MAGNETIC BUBBLE MEMORY
AS SECONDARY STORAGE FOR
MICROPROCESSOR LABORATORY

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

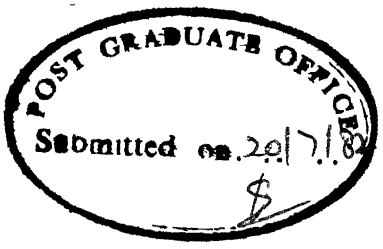
VINOD P. DESHMUKH

to the
DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY, 1982

– dedicated to my sisters

VAIJU and MEENA

with great love



CERTIFICATE

This is to certify that the thesis entitled MAGNETIC BUBBLE MEMORY AS SECONDARY STORAGE FOR MICROPROCESSOR LABORATORY by Vinod P. Deshmukh has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

R Biswas

(R.N. Biswas)
Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur.

ACKNOWLEDGEMENTS

I am greatly indebted to my thesis advisor Professor R.N. Biswas for his able guidance and enlightened comments throughout the course of this work. It has been altogether a different experience to have worked with Dr. Biswas and I would like to thank him for the helpful suggestions and numerous discussions I had with him at various stages of this study.

I am thankful to Mr. Prem Malhotra, Research Engineer, ACES for the co-operation he has extended to me during this period.

My friend Govind needs special mention for the selfless nature of his help rendered to me on several occasions. Finally, I would like to acknowledge the help and encouragemen received from my friend SUK Pillai who has made my stay here a memorable occasion.

The neat, skillful typing of Mr. C.M. Abraham is appreciated.

Vinod P. Deshmukh

ABSTRACT

An attempt has been made to design the secondary storage for a multiterminal multiuser laboratory used for training people in the area of microprocessor applications. Out of the many available alternatives for the storage device, magnetic bubble devices have been chosen for their compactness and cost-effectiveness for the system envisaged. A file management scheme based on linked-block and file map techniques has been proposed and implemented with the Intel 8085 microprocessor as the CPU. The scheme has the advantage of direct access to pages while minimizing the wastage of secondary memory owing to unused portion of the pages allocated to file maps. An 'appropriate' coding scheme, for the application in question, has been implemented in software to take care of burst errors occurring during the transfer of data through the interface link. A secondary storage of 1 Mbyte capacity has been designed and the system has been successfully tested with 128 Kbytes of secondary storage.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The growing need for training people in the microprocessor field has made educators think of developing specialized laboratories capable of handling large numbers of students. Any such microprocessor laboratory should be structured to support graduate research using microprocessors as well as to provide means for introducing microprocessor applications to undergraduate students. The laboratory facility should provide a high level of software and hardware support to the microprocessor activity. For hardware interface, functional microcomputer modules should be available for integration into a microcomputer system. The key feature of these modules is a standard input/output structure which would make them compatible with standard semiconductor memories, peripheral mass storage devices, computer terminals, analog input and output devices etc., and, of course, with various microprocessors.

The primary characteristics of a microprocessor laboratory should be a multiterminal facility so as to give services to many users at a time. The total number of users may be 4 to 5 times the number of users that can use the laboratory facility at a time. The laboratory may have to support typically 30-40 undergraduate students, be an

experimental facility for 5-10 graduate research projects, and may sometimes also have to be a medium for carrying out 5-10 independent projects. The number of persons over an academic session may thus be about 60.

Other constraints on the design of the laboratory come from the environmental conditions, economy and the type of users. Economic constraints limits the expansion of the system, in particular the size of the secondary storage.

1.1 OVERALL ORGANIZATION OF A MICROPROCESSOR LABORATORY

A schematic representation of the organization of a multi-user microprocessor laboratory is shown in Fig. 1.1.

The laboratory is centred around the system controller, the design of which has been one of the main concerns of this study. It is a means for file organization and provides operating system support in some sense. Secondary memory is for mass data storage and is controlled by the system controller. It is shared by all the users of the laboratory. There is a number of video intelligent terminals in the system, having facilities to edit the file, execute the program, and to communicate with the system controller. It has an additional hardware attachment facility which provides a means for its users to test a wide variety of modules. The printer provides a facility to take a hardcopy of any data.

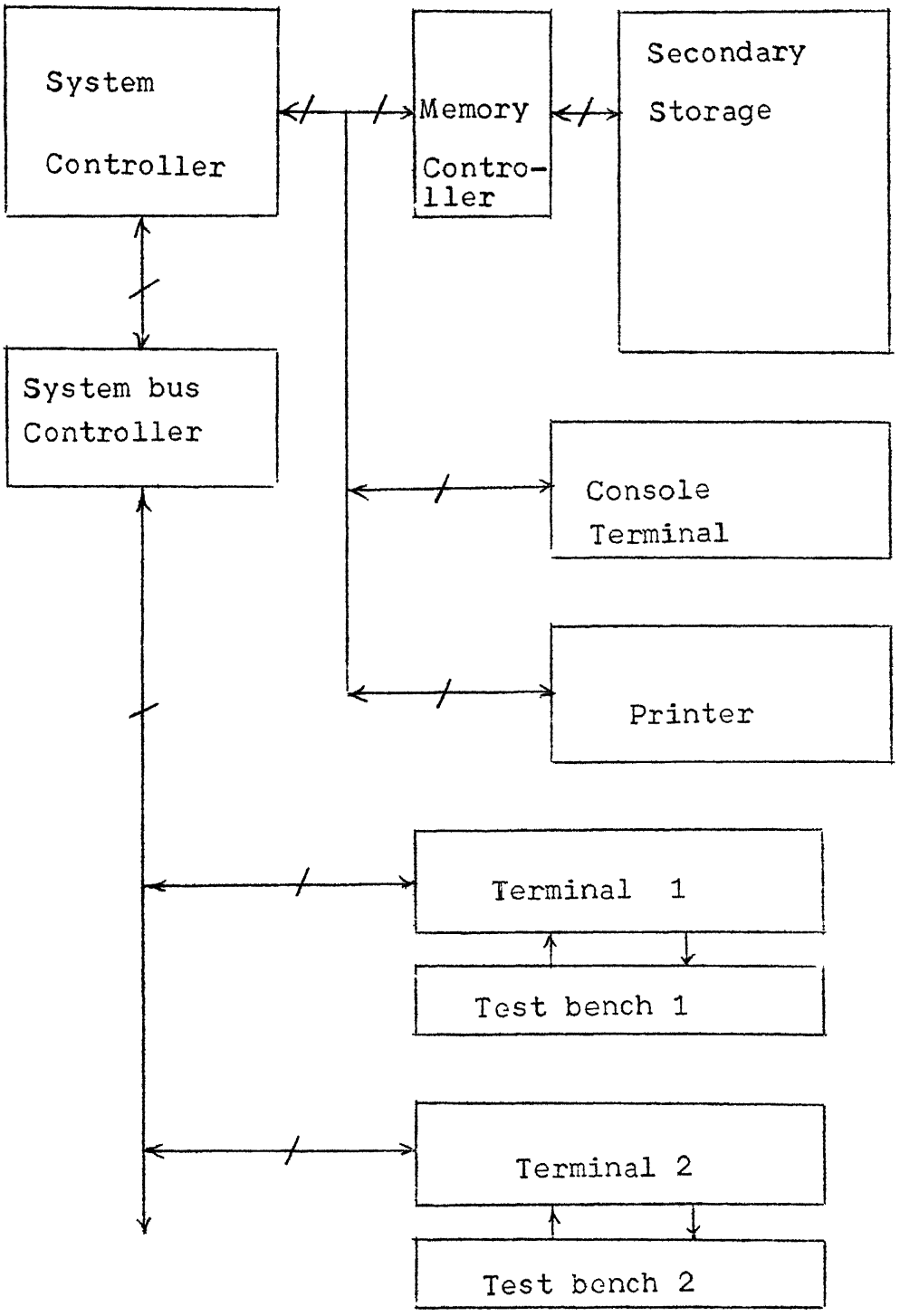


Fig. 1.1 Organization of Microprocessor Laboratory

1.2 SELECTION OF SECONDARY STORAGE DEVICE

On an average, 16 Kbytes of secondary storage is enough for the kind of user who will make use of this laboratory. Assuming the maximum number of users having access to these facilities to be 64 (a convenient power of 2), it is seen that the required size of the secondary storage is 64x16 Kbytes = 1024 Kbytes (= 1 Mbyte).

Different type of devices used for secondary storage can be categorized as follows :

1. Solid state memories -
   - i) charge-coupled devices,
   - ii) magnetic bubble memory devices (MBM)
2. Rotating mass memories -
   - i) cassette tape recorder,
   - ii) Floppy-disc storage devices,
   - iii) Moving-head disc,
   - iv) fixed-head disc.

The rotating mass memory systems require controlled humidity and dust free environment. These conditions demand an airconditioned clean room for the installation of such systems. Also these systems are sensitive to power line fluctuations. On the other hand, solid state memory systems operate on dc power supply, which can be easily stabilized against fluctuations in the mains voltage. As operating

temperature for the solid state devices ranges from $0^{o}C$ to $65^{o}C$, there is no need to maintain airconditioning (the environmental temperature is normally within the operating range of the device).

Moreover, solid state memory devices, being small in size, have packaging advantage over the rotating mass memories. Since few megabits of the storage can come on a single printed circuit board, this also solves the problem of housing the secondary memory using solid state devices easily.

Table 1.1 presents a comparison among the different mass storage devices and Fig. 1.2 shows relationship between cost per bit and storage capacity for different memory devices. It is evident from Fig. 1.2 that the cost per bit of solid state memories is almost independent of the storage capacity, while that of rotating magnetic memories is highly dependent on the storage capacity, and falls rapidly with increasing size. As a result, for applications that do not require a high capacity of secondary storage, solid state memories will be more cost-effective.

Among solid-state memory devices, CCDs require a power back-up for permanent storage of data and, as such, are not suitable for secondary storage applications in general.

Thus considering the factors like relative cost, ruggedness, need for maintaining environmental conditions, ease

Table 1.1

Current Computer Mass Memory Technologies Review

General Characteristics	CCD	MBM	CASSETTE	Floppy Dis	Moving Head Disc Cartride	Fixed head dis
Access time	100 µsec.	1-3 msec.	40 sec.	300 msec.	60 msec.	8 msec.
Transfer rate	1M to 5M bits/S	100Kbits/ sec	10Kbits/ sec	250 Kbits/ sec.	1.5Mbits/ sec.	4Mbits/sec.
Storage Capacity(typ)	64Kbit per chip	128Kbit/ chip	4Mbits	3.1Mbits	24Mbits	4Mbits
Error rate	1 in 10^{10}	1 in 10^{12}	1 in 10^{7}	1 in 10^{8}	1 in 10^{10}	1 in 10^{10}
Reliability(MTBF in hrs)	7000 +	10,000 +	3000	4500	4500	10,000
Removable Media	No	No	Yes	Yes	Yes	Yes
Nonvolatile	With battery	Yes	Yes	Yes	Yes	Yes
Soft-ware interfacing capability(1=easiest)	2	3	4	5	6	5

contd ..

contd...(Table 1.1)

General characteristics	CCD	MBM	CASSETTE	Floppy Disc	Moving Head Disc Cartride	Fixed head disc
Usable bits	2 Mbits	3Mbits	3Mbits	2Mbits	20Mbits	3.3Mbits
Transfer rate	2Mbits/sec	0.8Mbits/sec	10Kbits/sec	250Kbits/sec	1.5Mbits/sec	4Mbits/sec
Power (W)	30	30	25	75	150	350
Weight (lb)	4	8	8	25	60	100
Size (in)	PC board in CPU	PC board in CPU	5x5x5	5x14x10	18x23x7	19x19x12

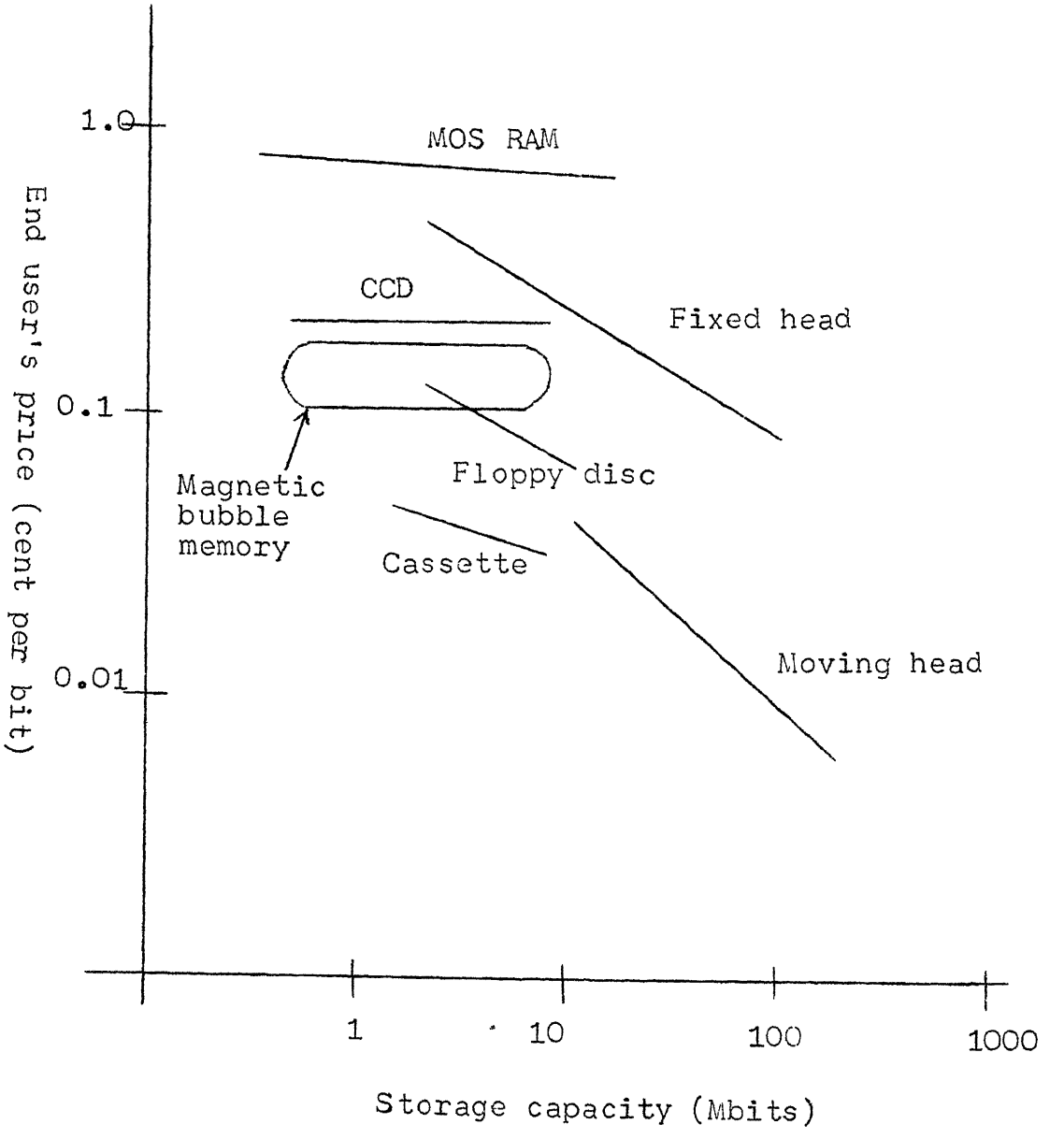


Fig. 1.2 Storage capacity vs price per bit

in packaging and nonvolatility, we see that MBM is the best secondary storage device in this particular situation.

However, most established business houses employ rotating mass storage devices for secondary storage today, and it is certainly not possible for them to switch over immediately to secondary storage using MBM even for relatively low-capacity requirements, in view of their existing facilities and the investments made therein. But for development of new systems it will definitely be advantageous to use MBM for low capacity secondary storage. This project aims at the development of a 1-Mbyte secondary storage system using MBM.

The organization of the thesis is as follows :

Chapter 2 gives a brief description of the technology used for magnetic bubble device,and the techniques used for bubble access. It also describes the memory board and some operations on the magnetic bubble memory.

Chapter 3 starts with a discussion on the different types of information management schemes and their shortcomings. A scheme for our specific application is then proposed, and the chapter ends with implementation details of the proposed scheme.

Chapter 4 deals with the system controller -- the most important block of the microprocessor laboratory. It includes different tasks that the controller has to perform, the flow charts for the different commands that it has to obey and finally the hardware description of the CPU board.

Chapter 5 describes the need for a coding scheme for transmission of data. Taking into account that errors occur in bursts, a suitable scheme with interleaving is proposed. It also includes the implementation details of the coding scheme.

Test results of hardware and software are reported in Chapter 6.

CHAPTER 2

MAGNETIC BUBBLE MEMORY

Bubble memories can be viewed as a solid state version of rotating magnetic memories, such as disc and tape recorders. In both these types, information is stored in the form of magnetized regions. These regions, in the integrated version, are cylindrical domains, the presence of which at a particular location corresponds to a binary digit at that location. These bits are made accessible by moving the domain to an access device as opposed to moving the storage medium as in the case of disc or tapes.

Following sections give review of the state-of-art of the bubble technology. Sections 2.1 and 2.2 review the basics of the magnetic bubble devices and the magnetic bubble memory mechanisms. In Section 2.3, the actual organization of commercial bubble memory devices is described in some detail. Finally, Section 2.4 describes the magnetic bubble memory system.

2.1 MAGNETIC BUBBLE DEVICES

The formation of magnetic bubble in certain magnetic thin films was first reported by Bobeck et al. [6] in 1966. A bubble is nothing but a cylindrical magnetic domain having magnetization opposite to that of the surrounding material in a magnetic thin film. This phenomenon is observed under the

action of an applied magnetic field in certain thin film materials,e.g., epitaxially grown garnet, which possess magnetization characteristics with the easy direction of magnetization perpendicular to the plane of the film. Fig. 2.1 shows the different configurations of magnetic domains in such a film under various values of applied magnetic field.

In the absence of any external field, the film plane is divided into two regions of interwined serpentines, one with upward magnetization and the other with downward magnetization (Fig. 2.1a). When a bias field is applied normal to the film plane, the regions with magnetization parallel to the field will grow at the expense of the other region (Fig. 2.1b). As the external field increases the serpentines of the latter region continue to shrink and ultimately get reduced to cylindrical domains called 'bubbles' (Fig. 2.1c). With further increase in magnetic field the diameters of the bubbles go on decreasing upto a certain critical field, beyond which the bubbles 'collapse', i.e., the magnetization of the film becomes uniform. Fig. 2.2 shows how the diameter of the bubbles varies with the change in the applied magnetic field within the range, bounded at the lower end by 'strip out' (serpentine domains) and by 'bubble collapse' (uniform magnetization) at the other end.

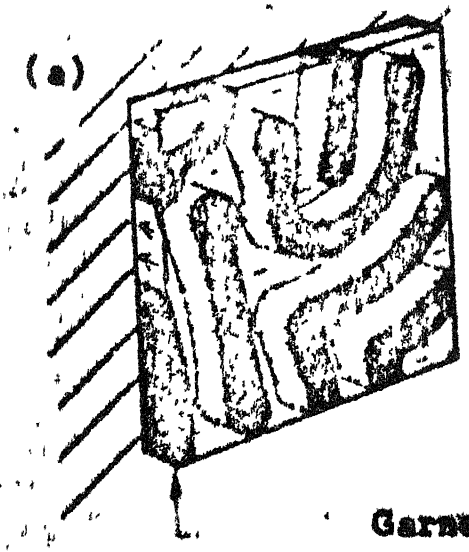


Fig. 2.1 Different configurations of the magnetic domains under applied magnetic field

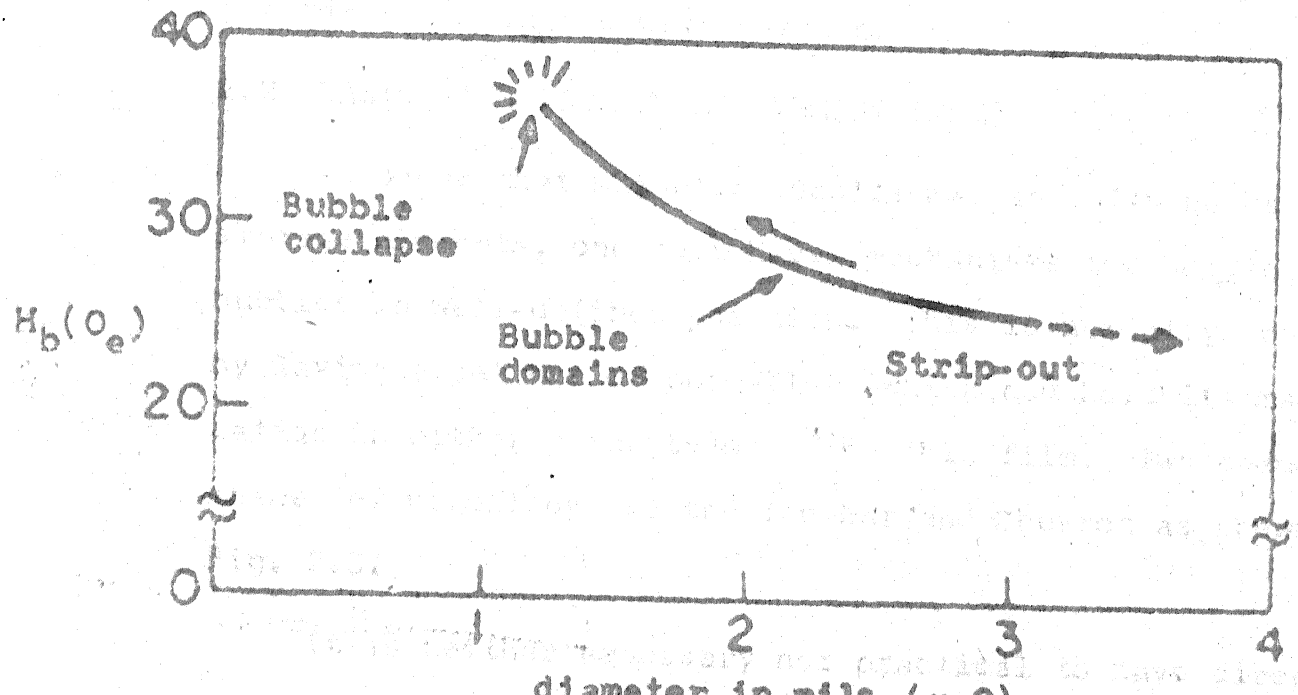


Fig. 2.2 Diameter of magnetic bubbles as a function of an applied bias field

One notes that the diameter of the bubbles remain stable in the range from 3 mils to 5mils for a typical range of applied magnetic field from 30 Oe to 25 Oe. It is thus possible to have a million bubbles within an area of one centimeter square. As each bubble can represent a bit of information, the potential of magnetic bubbles for highly compact mass storage is evident. The actual structures of such memory devices and the techniques used for using them are discussed in the next section.

2.2 BASIC MAGNETIC BUBBLE MEMORY (MBM)

In order that magnetic bubbles may actually be used as storage elements, one must have mechanisms for holding the bubbles in well-defined position. This is normally achieved by laying a pattern of permalloy (which can hold its magnetization in either polarity)over the thin film. Two common shapes of permalloy are the Tee-bar and Chevron as shown in Fig. 2.3.

It is neither necessary nor practical to have direct access to all the storage elements in a mass storage device. The MBM is, therefore, organized on the basis of serial access by appropriate configuration of the permalloy patterns, all the bubble sites within this pattern forming a single loop, equivalent to a single track in a rotating magnetic storage.

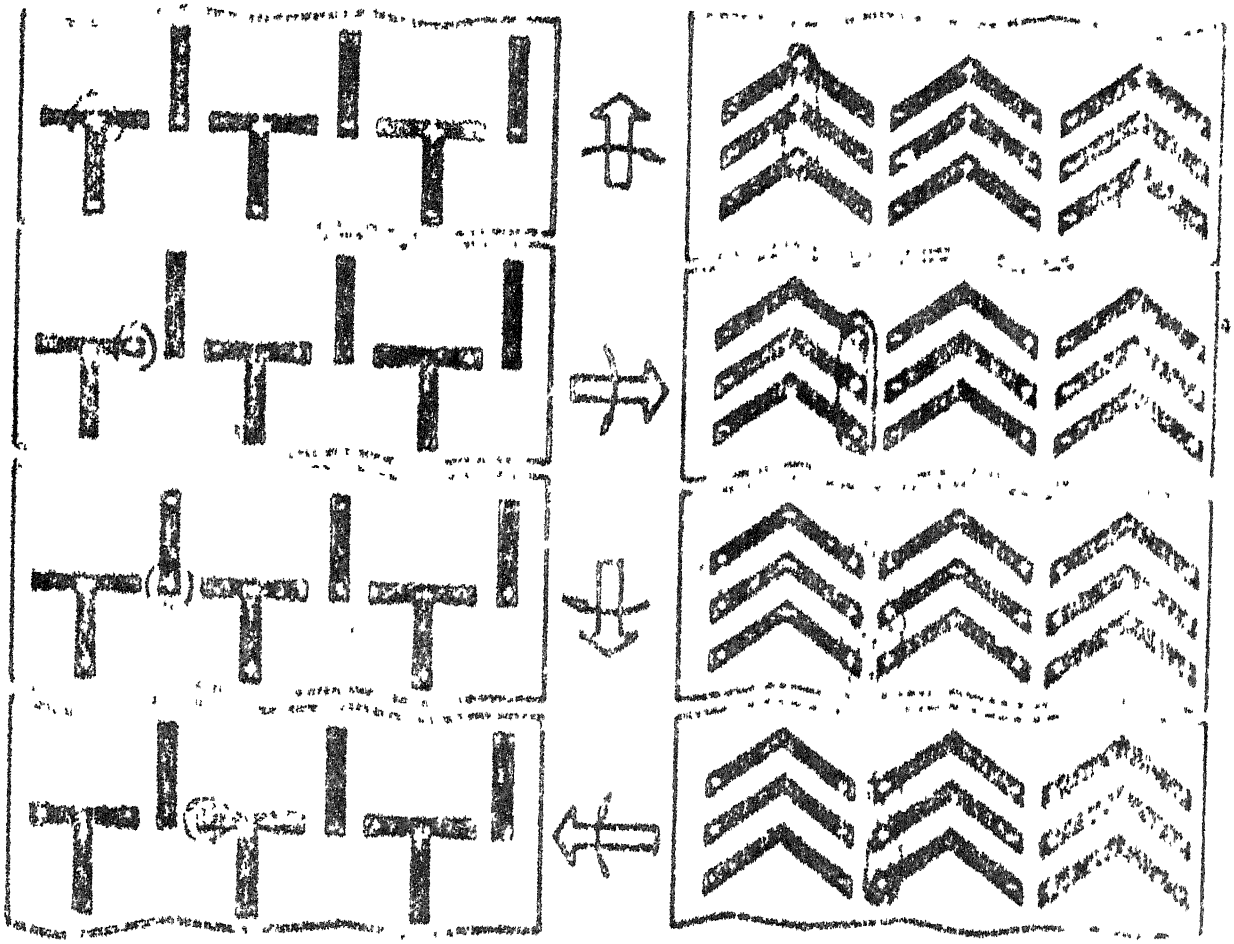


Fig. 2.3 Propagation of bubbles in Tee-bar and Chevron patterns

Fig. 2.4 shows the organization of such a loop.Bubbles can be generated only at one point in a loop labelled as the GENERATE gate and can be sensed at a second point called the DETECT gate. The analogy with a conventional shift register connected as a ring is evident. Clearly, in order to read from or write into any locations the bubbles must be made to propagate along the loop until the desired bubble reaches the appropriate gate. This requires a mechanism for propagating the bubbles in addition to the mechanisms of generating and detecting them.

2.2.1 Propagation of Bubbles

The propagation of bubbles is guided by the permalloy pattern by rotating the field in the permalloy. Fig. 2.3 shows the propagation of bubbles in chevron pattern. Here the bubble is considered as a negative pole which gets attracted to a pole having opposite polarity formed in the permalloy layer. The magnetic field is provided by two coils. By controlling the current through these coils, a field in any direction in the plane of the permalloy pattern can be generated. For propagation of bubbles, the magnetic field is made to rotate continuously. Parts a,b,c and d of Fig.2.3 show successive magnetizations in the permalloy and hence the positions where the bubbles are held after the first, second, third and fourth quarters of a complete cycle of rotation of the field. Every bubble thus advances by one

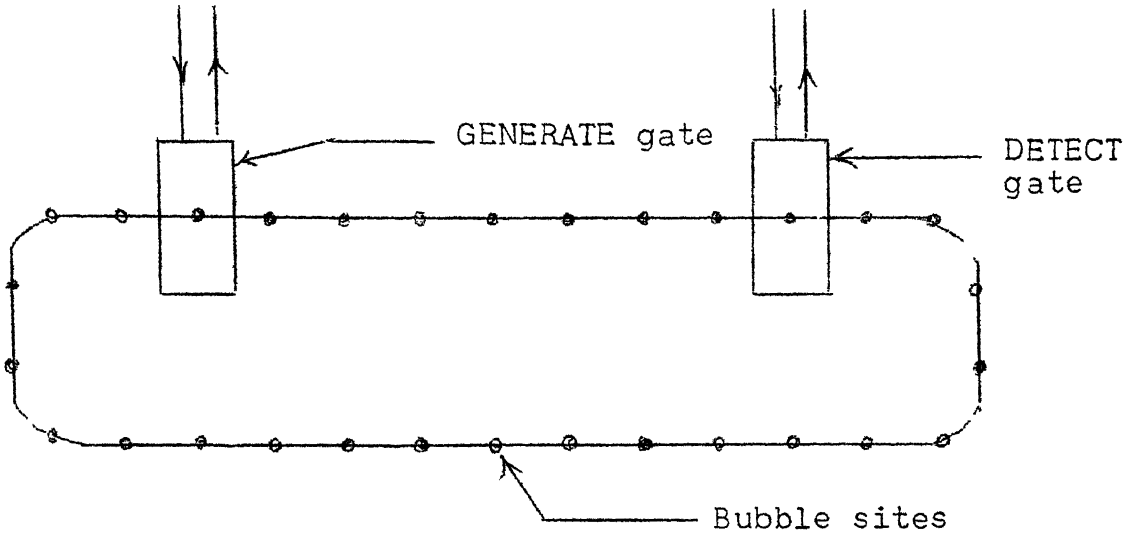


Fig. 2.4 Single shift-register organization of magnetic Bubble memory

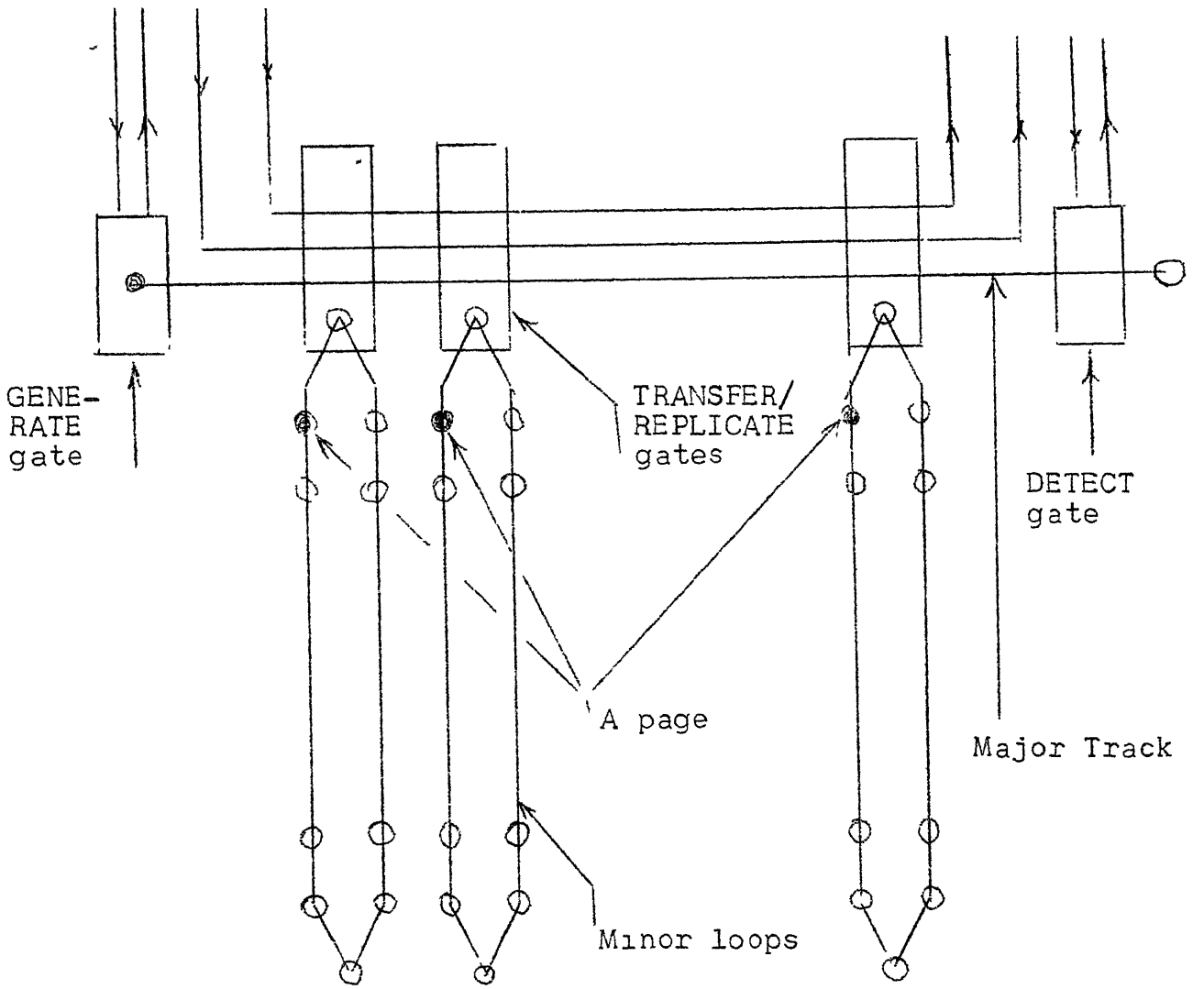


Fig. 2.6 Major-track/Minor loop organization

chevron period for each complete rotation of the field.

2.2.2 Generation of Bubbles

Fig. 2.5 shows, schematically, the process of generation of bubbles. Here, through a conductor loop, patterned near the epitaxial layer a current pulse is passed. The current in the loop causes large localized magnetic field which reverses the magnetization of the epitaxial layer at that point, i.e., forms a bubble.

2.2.3 Detection of Bubbles

A bubble is detected by exploiting the magneto resistive property of certain varieties of permalloy. The detector is used as one of the arms of a bridge. This produces a voltage change of about 5-10 mV at the detector terminals when a bubble passes under the detector. The bubble is stretched to its maximum length before passing through detector to achieve maximum efficiency of detection.

2.3 MAJOR-MINOR LOOP ORGANIZATION

The simple shift register organization, described in Sec. 2.2, has the drawback of large access time, and hence an analogous arrangement to multiple track rotating magnetic storage, called major-minor loop organization is more commonly used.

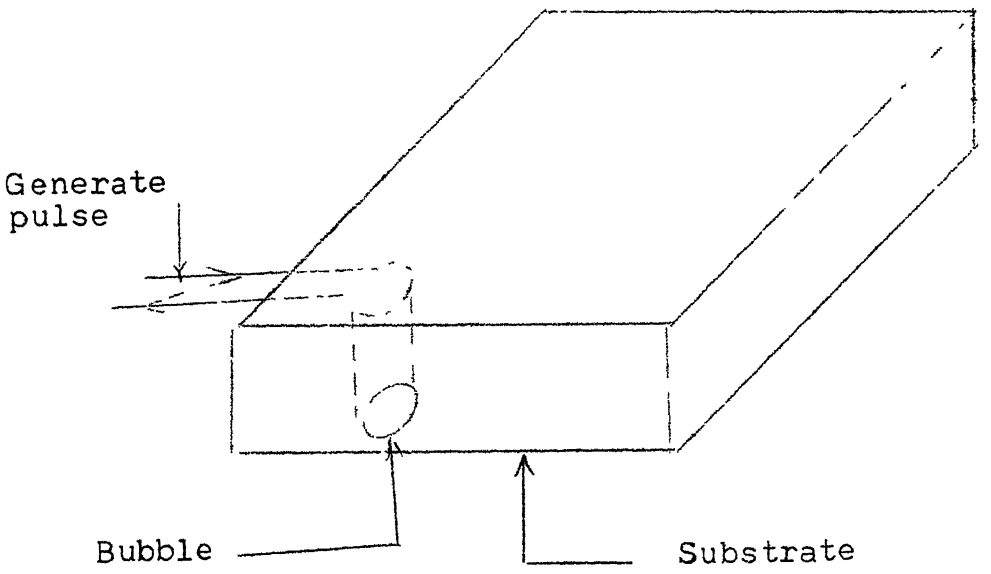


Fig. 2.5 Process of generation of bubbles

In the major-minor loop organization, information is stored in many parallel closed shift registers called the minor loops, where data is circulated. When read or write operation is to be performed the access to the minor loops is provided by a common shift register. The common shift register can either be a closed loop or an open track. If it is a closed loop then an additional function of clearing the loop is necessary, whereas, if it is open track, clearing can be achieved just by expelling the bubbles from the track. M/s. Intel Corporation has chosen the open track, called the major loop, arrangement for their bubble memory organization shown schematically in Fig. 2.6. The major track links the minor loops as well as the read and write circuitry. Here, in addition to the GENERATE and DETECT gate, there is a need of transferring bubbles from the major track to the minor loops and vice-versa. This is achieved by TRANSFER/REPLICATE gates. One such gate is provided for each minor loop. The mechanisms of the TRANSFER/REPLICATE gates are explained later.

In this organization, all the bits in the device are circulated in synchronization with the rotating field, as a result of which the positions of all the bits are determined by the clock timings. The bits occupying corresponding positions in all the minor loops (one bit per minor loop) constitute a 'page'. As can be seen from Fig. 2.6, one page is

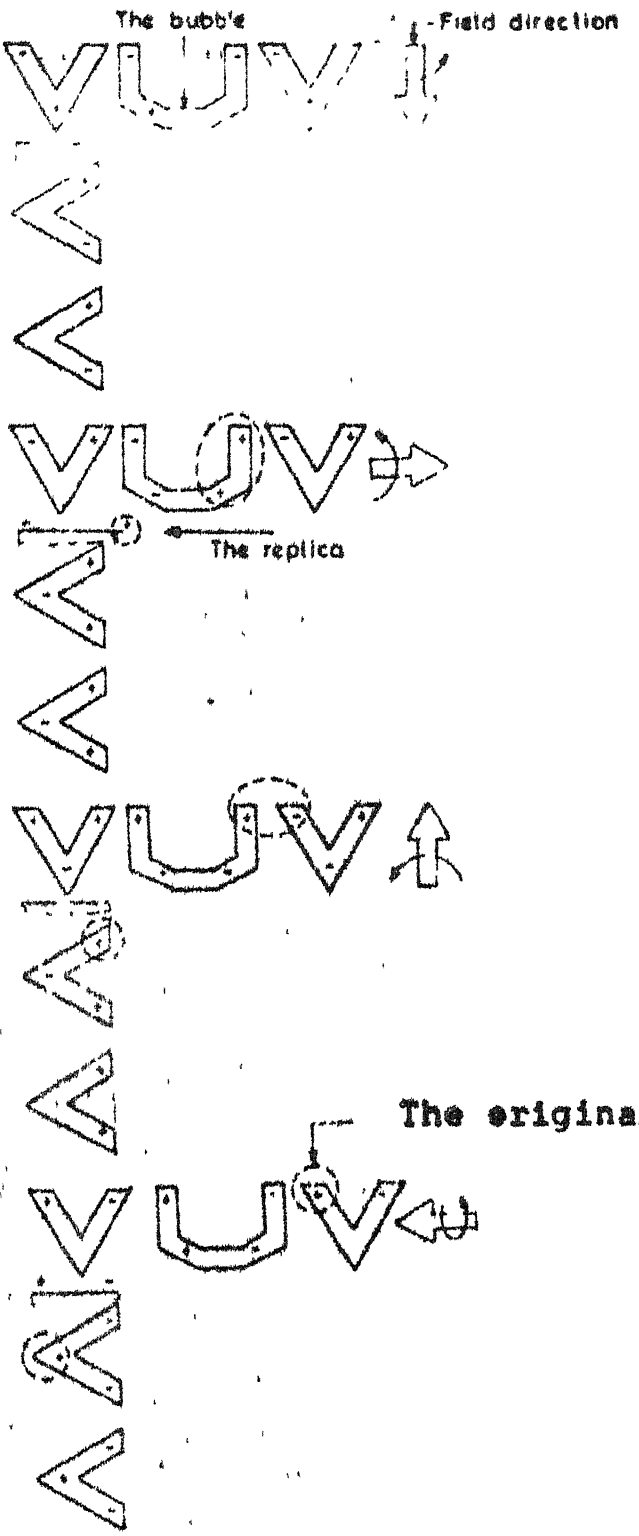
transferred, at a time from the minor loops to the major trackor vice-versa.

For reading a page, the page is moved to the TRANSFER/REPLICATE gate adjacent to the major track. All the bits from the page are duplicated in parallel on the major track. After a fixed delay, the first bit of the duplicated page will be at the DETECT gate. The bubble passes through the detector and after it is sensed gets expelled from the major track. For new information to be written, first the major track is cleared. The GENERATE gate is activated to insert data into the cleared major track in proper time phase. Then after properly aligning the major track and minor loops the TRANSFER/REPLICATE gates are energised to transfer data into the minor loops.

2.3.1 Replication and Transfer of Bubbles

Fig. 2.7 shows a possible configuration of the REPLICATE/TRANSFER gate. For nondestructive read out, a bubble is moved to a gate where it is stretched and at a proper phase of the rotating field, two successive current pulses are applied, first to cut the bubble into two and then to transfer it. With the rotating field the original bubble continues to move along the same path, while its replica moves on along the track/loop to which it has been transferred.

The same gate also serves as a TRANSFER gate. In this mode instead of the cut pulse and the transfer pulse, only the



Cut pulse at this instant will cut the bubble into two (just for transfer operation a high transfer pulse is applied at this instant).

A transfer pulse at this instant will transfer the bubble to other track

Fig. 2.7 A possible configuration for REPLICATE TRANSFER gate

transfer pulse is applied at an earlier instant. As a result, the bubble follows only the replicated bubble path.

2.4 MAGNETIC BUBBLE MEMORY SYSTEM

Along with the bubble memory device, as seen in last Section, there is need for support circuitry to perform the following functions.

i) Precise current pulse generation to generate current pulses at appropriate instant and of proper shape for generation, duplication, transfer or replicate functions.

ii) To sense the low level analog voltage from the detector.

iii) To generate both current waveforms in the drive coil to get the rotating field, and

iv) to generate all the control signals for coordinating the timing of these operations.

Fig. 2.8 shows the block diagram of basic magnetic bubble memory system.

For each of these functions there is a separate chip provided by Intel Corporation.7110 is the magnetic bubble memory device with 1 Mbits active storage. It is organized in a serial-parallel-serial shift register fashion as shown in Fig. 2.6. The data is organized as a 512-bit page with a total of 2048 pages. The 512 bits are divided into two

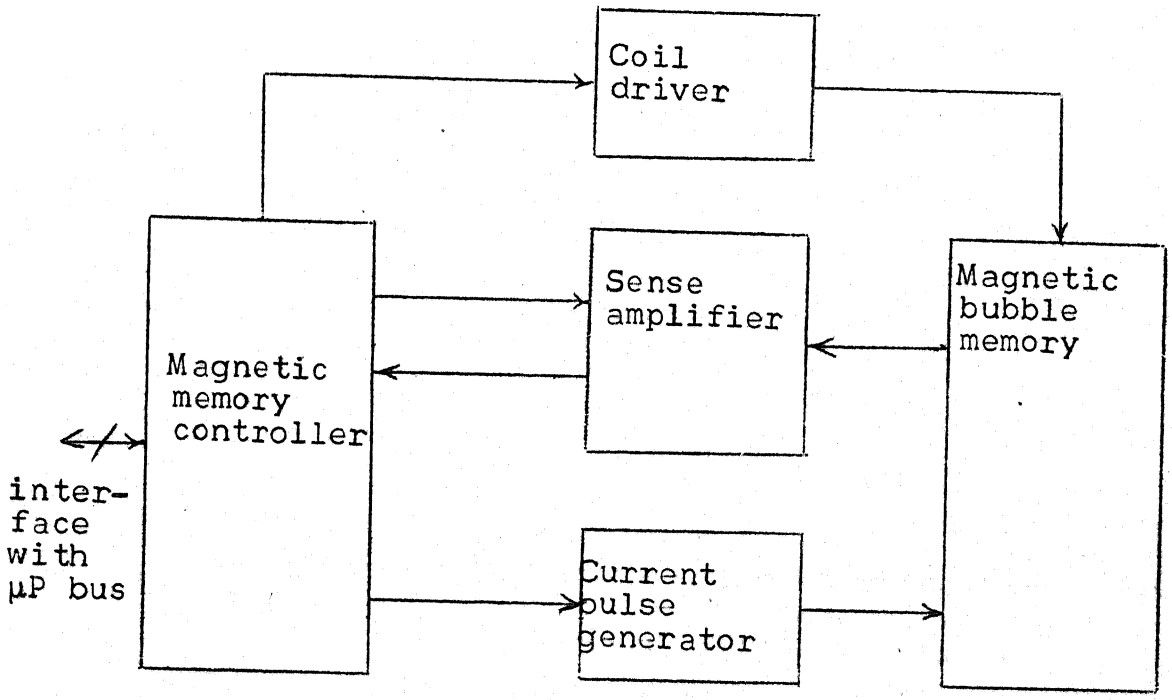


Fig. 2.8 Block diagram of Magnetic Bubble Memory System

channels of 256 bits each to double the data rate. There are 64 additional loops so that defective loops can be isolated during testing, thus increasing the chip yield. The map of the good loops is stored in a loop within the bubble memory and is called the boot-loop. This map is copied in the formatter and sense amplifier (7242) when the system is put on. All the good loops are not necessarily used. There is a provision to change the boot-loop map so that even if any of the good loops goes bad during operation, it can be masked. The formatter and sense amplifier helps avoiding the data from entering into the spare loops with the help of the boot loop map.

The 7220 is used as the bubble memory controller. At one end it can be directly interfaced with any microprocessor and depending upon the commands it issues the different control signals to the sense amplifier (7242), coil driver (7250) and the current pulse generator (7230), which act on the bubble memory device to execute the command. Data transfer between controller and sense amplifier is on a serial line.

CHAPTER 3

INFORMATION MANAGEMENT

For small microprocessor-based systems secondary storage is one of the important considerations. Total expenditure, reliability and environment suitability are vital factors in deciding the type of the secondary storage to be used. As discussed in Chapter 1, magnetic bubble devices are chosen as the secondary storage devices for this project. These are now comparable in cost with floppy-disc storage, and the trend in prices indicates that, in the near future, magnetic bubble devices will supersede floppy-disc storage devices.

In our multiuser multiterminal system, discussed in Chapter 1, one of the important problems is that of managing the secondary storage. In such a system the volume of data to be handled is normally small. This together with considerations arising from the cost, limits the size of the storage. Because of the relatively small size of the secondary storage, every care should be taken to utilize the memory space to its maximum capacity and avoid problems arising from the fragmentation of memory or duplication of data in the memory.

The next section gives a review of different information management schemes (IMS), with the resulting requirements for the secondary storage. In the following section an IMS suitable for the present problem is evolved and the implementation details of the scheme are presented.

3.1 INFORMATION MANAGEMENT SCHEMES

A user will need a facility to record some data permanently under some identifier. The user should be able to write, read or delete the particular record using the same identifier. Such a record of data is referred to as a 'file' and the identifier as the 'file name'. For sake of user's convenience each file is generally visualized as a sequence of 'pages', each page representing a block of record in the actual storage device. In most memory systems, the size of a block and hence that of a page is uniform, mainly due to hardware considerations. In all further discussions, therefore, the words 'page' and 'block' will be used interchangeably to indicate a fixed unit of record.

An information management scheme should be in a position to provide the facilities to read, write or delete files. In addition, the following points are to be considered for a good information management scheme.

i) Different users may accidentally choose the same name for a file which should be distinguishable.

ii) In order to avoid duplication of data, sharing of the files is necessary, and as such the IMS should have a procedure to decide the right of access to the files.

iii) The IMS should be able to take care of files having variable length.

iv) The IMS should, as far as possible, be independent of the secondary devices actually used for storage.

The efficiency of the scheme mainly depends on how the files are stored, as well as on whether there is any fragmentation due to deletion of files.

Depending upon the information management scheme, one or more directories may have to be maintained for the files. The entries in the directory consist of the name, the access right, the starting address and the size for each file.

The following subsections describe, in brief, the schemes normally used for the management of information. Shortcomings associated with each of these schemes, that should be overcome in a good IMS, are also pointed out.

3.1.1 Simple File System

Here the file is structured from the user's point of view as a sequential file. It is also physically stored as a sequential file, i.e., the entire file is stored in consecutive

physical locations. The physical location of the file on the secondary device is determined by means of the starting block number and the number of blocks stored in the directory. The shortcomings of such a simple file system are as follows.

(a) The entire file directory can not be stored in the main memory as it requires a large amount of memory. The other method is to store the directory as a file on the secondary storage itself but then it takes a substantial amount of time to access any file.

(b) Each file can have only one name. It restricts the user to use the same name for a particular file.

(c) For each file big access and protection tables are to be maintained.

(d) Files are stored contiguously on the secondary storage and so fragmentation is bound to occur after the deletion of some files i.e. such a scheme does not use the secondary storage effectively.

3.1.2 Symbolic File System

The function of the symbolic file system is to map the user's symbolic references to suitable identifiers. In this scheme the file directory is divided into two components :
i) symbolic file directory concerned with file naming and
ii) basic file directory concerned with the physical file.

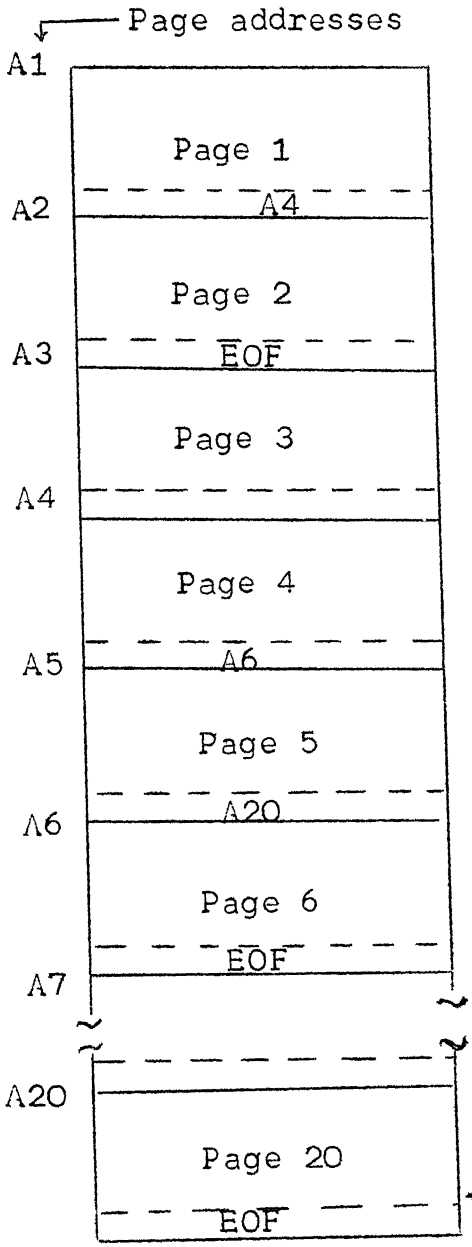
This system allows multiple names to be given to the same file, but does not allow logical segment size independent of physical block size; moreover, noncontiguous allocation of memory space is not possible. These shortcomings are overcome in the next scheme described.

3.1.3 Physical File System

The primary function of the physical file system is to perform a mapping of logical byte addresses into physical block addresses. In this scheme physical blocks of the file may be noncontiguous. The popular techniques to achieve this are (i) chained blocks and (ii) file map.

In the chained-block approach, depicted in Fig. 3.1, each physical block contains the address of the next 'logically contiguous' block as an 'address pointer'. The basic file entries contain the address of the first physical block but subsequent block addresses are found by means of the address pointers. In this scheme, the address pointer is stored within the physical block itself, thereby reducing the capacity of the block. This type of chained-block approach is efficient for sequential access to the block but quite inefficient for direct access.

In the file-map technique, on the other hand, the address of all the blocks of a file are stored in a separate block, called the file map, as pictorially represented in Fig. 3.2. This permits direct access to any block in a file, the price



FILE	PAGES
File 1	Page 1 Page 4 Page 6
File 2	Page 2
File 3	Page 5 Page 20

Fig. 3.1 The chained block structure

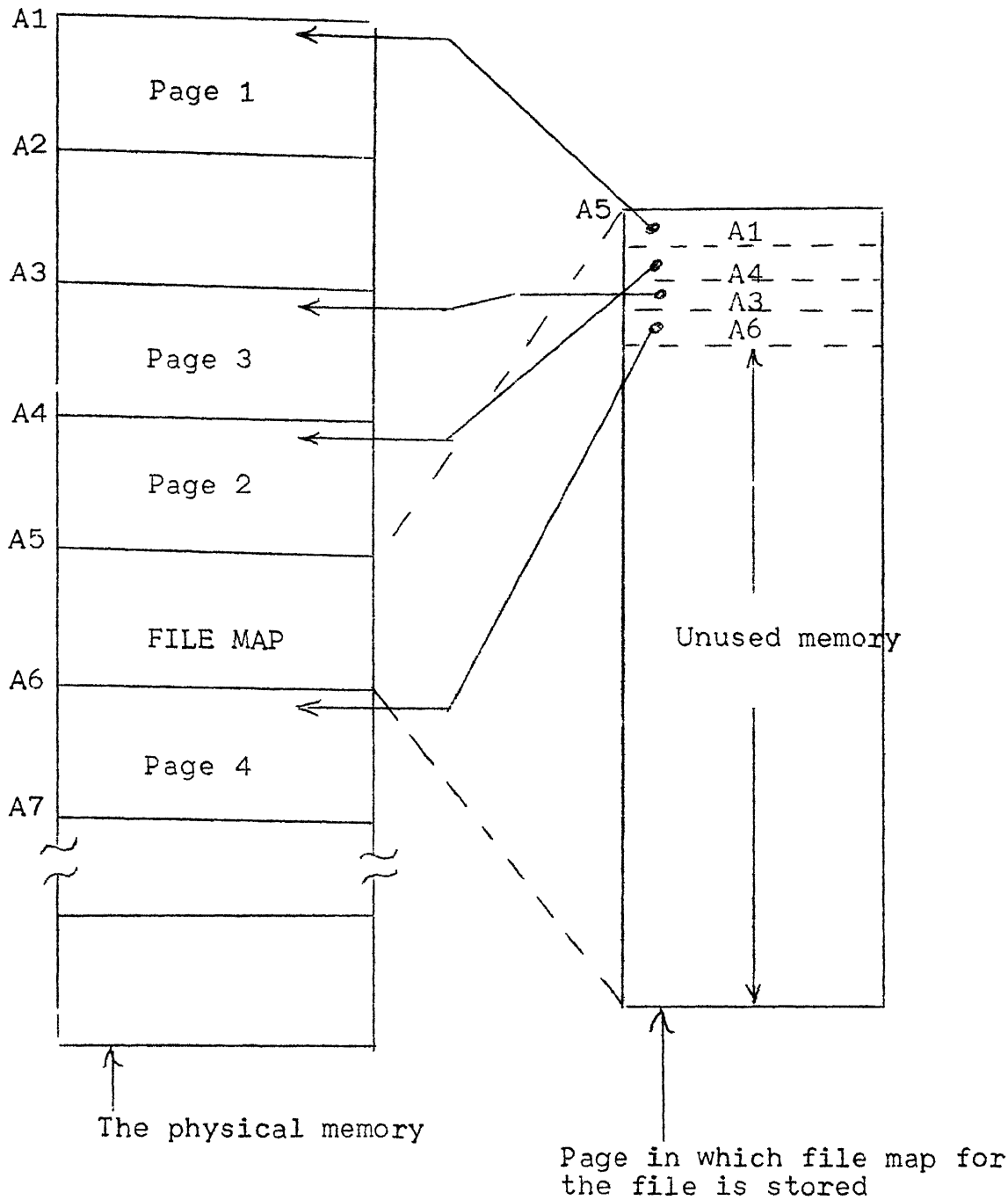


Fig. 3.2 The file map technique (shown for one file only)

that one has to pay for this advantage is two-fold :

i) For very large files, the file map will exceed a block and hence will, in turn, require a file map again.

ii) For small files, the file map unnecessary wastes an entire block.

3.2 THE PROPOSED SCHEME

It is clear from the foregoing review that a physical file system is our best choice. However, it has also been noted that each of the two types of physical file systems discussed still suffers from shortcomings which are, in some sense, complementary to each other. A combination of the two techniques is therefore proposed, as shown in Fig. 3.3.

This scheme is essentially a file map system with a chained structure for the file map, which is stored separately as a 'block table', and not as a page in the physical memory. With this approach it is possible to have direct access to any page in the file, just as in conventional file-map systems. But as the file maps for different files are contiguously stored in a common block table, this scheme reduces the unused memory in each page storing a file map (ref. Fig. 3.2). The price paid for this is the allocation of a fixed amount of memory for the block table irrespective of the number of files actually stored.

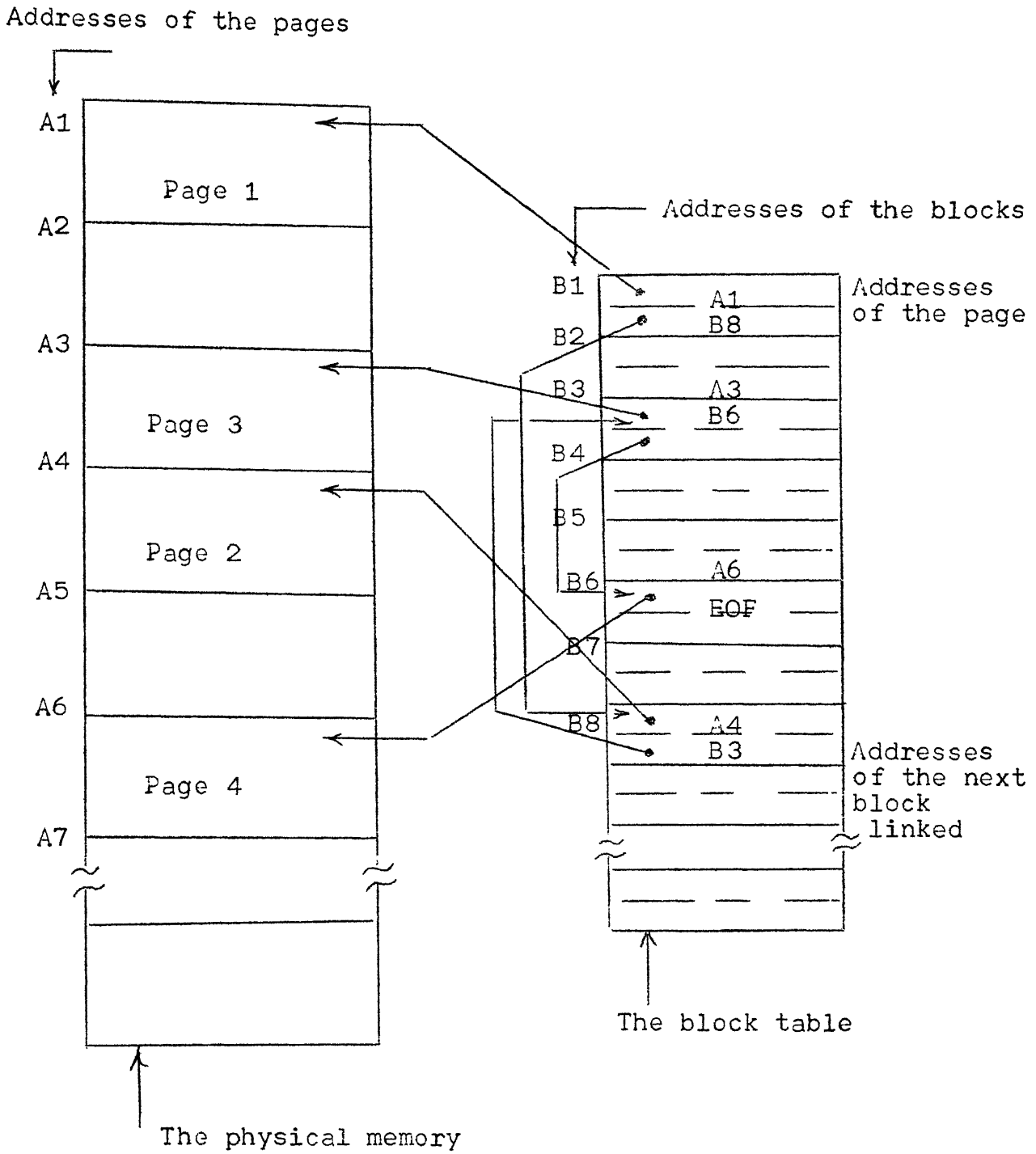


Fig. 3.3 The proposed scheme structure

In Fig. 3.3, each block consists of two addresses : first address is that of the current page in the physical memory, while the second address is that of the linked block pointing to the next page. The actual structure of these blocks will depend on the number of bits required for these addresses, which in turn, will depend on the choice of the page size. In view of the lower limit of 64 bytes on the page size (ref. Sec. 2. 4) inherent to magnetic bubble memory, 1 Mbyte MBM would require a maximum of the 14 bits for address for 64 byte pages. The decision regarding the actual page size will be taken up later in this section. However, it is evident that in a byte oriented structure as we have, each address will have to consist of two bytes, leaving a few bits as spares. One of the spare bits has to be used as the STATUS bit for the block, indicating whether the block has already been linked with a file or not. The byte structure of each block has thus been chosen as shown in Fig. 3.4.

3.2.1 Optimal Page Size

The effectiveness of the proposed scheme depends upon the size of the page. If the page size is too large then it leads to internal fragmentation. If size of the page is very small then it becomes necessary to have many block addresses which increases the size of the block table. We have, therefore, to evolve the optimal page size for the application at hand.

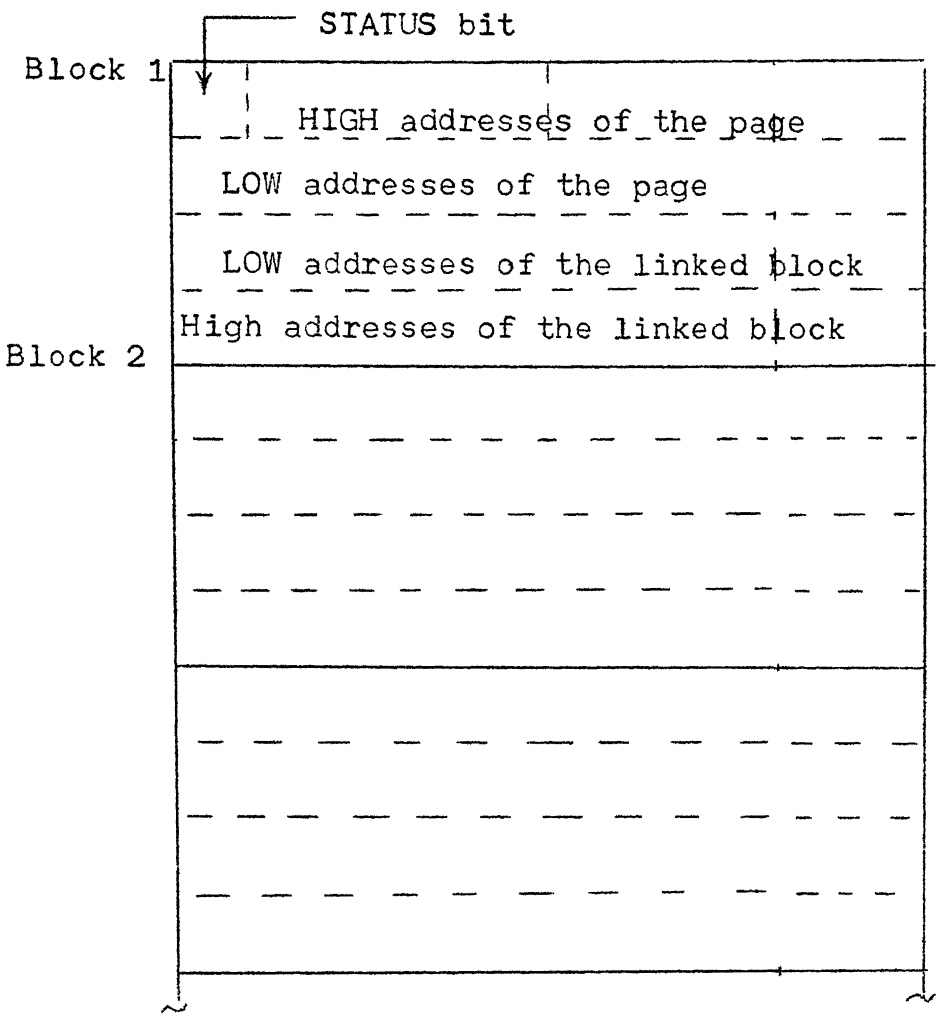


Fig. 3.4 The Block Table

For each page in the physical memory there are four bytes allocated in the block table [two bytes for the page address and two bytes for the linked-block address]. For an n-page memory, the block table will thus consists of 4n bytes. The separate storage requirement for the block tables is minimum if each block table corresponding to one user can be accommodated in a page. Table 3.1 shows for different choices of the page sizes, the maximum amount of memory that can be addressed by a user, if the block table size is restricted to one page. In the system visualized in Chapter 1, the average memory allocated to a user is 16Kbytes, and as such the maximum possible allocation to a user must be more; a page size of 256 bytes is thus inadequate. A page size of 1024 bytes is too large and in no case a user will be allowed to use 256 Kbytes of memory, which is 25 percent of total memory space. So the page size is chosen to be 512 bytes, which allows to allocate upto 64 Kbytes of memory space to a user. As the average size of the job in such a system will be around 512 bytes, internal fragmentation will also be minimum for this choice of the page size.

3.2.2 The File Directory

For facilitating the handling of files by the users, a table, called the file directory, has to be maintained for each user, containing the following four pieces of information.

Table 3.1

Page size in bytes	Max. no. of linked blocks in a page	Max. amount of memory that can be allocated to a user in Kbytes	Remark
256	64	16	Allocation too small
512	128	64	Optimum
1024	256	256	Allocation too large, Internal fragmentation

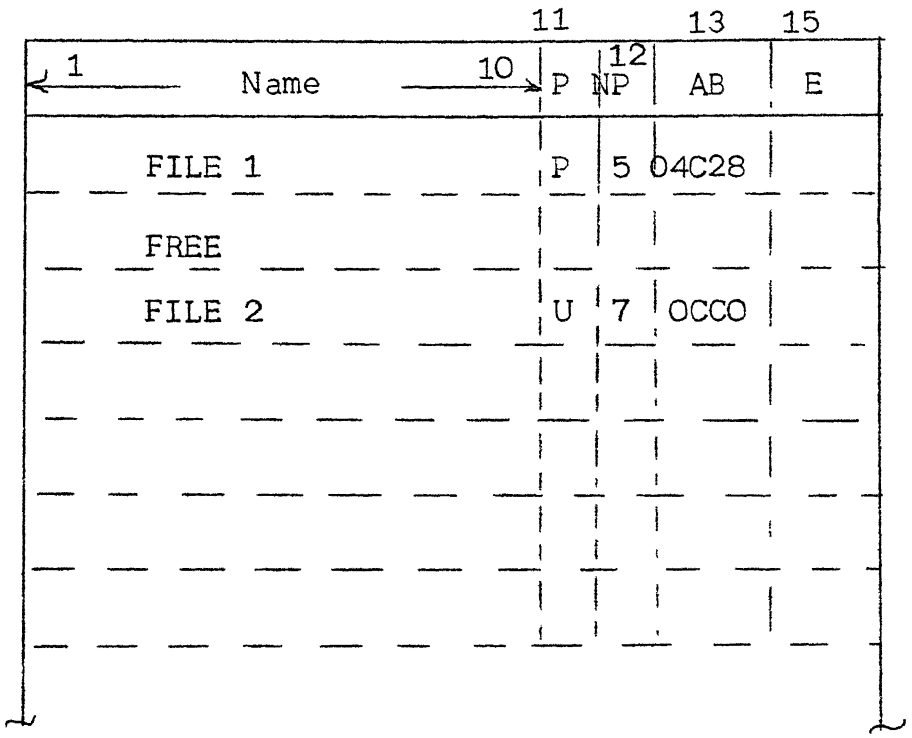
i) The name of the file : Most of the IMS generally use six character for the file name and three for the extension, with a special character in between. Thus in all ten characters are required for the name of the file. The same provision has been made in the present system, with the flexibility of extending the name itself upto 10 characters, if it is so desired.

ii) The access right : This decides whether a file should be allowed to be read or copied by any other user. One bit would have been enough for this field but as the organization is in byte form one byte has to be allocated. The additional bits may be used to incorporate any modification in the access policy.

iii) The number of pages : This information is necessary to know the length of the file. As the maximum number of pages that can be allocated to a user is 128, a single byte is sufficient to store this number.

iv) The first block address : This is the starting point of the linking process for each file. As pointed out in Sec. 3.2, it requires two bytes of memory.

From the foregoing discussion, we see that 14 bytes of storage are required for each file created in the file directory. Two additional bytes are left unfilled for any further modification in the scheme. The structure of the file directory is shown in Fig. 3.5.



N : Name of the file
P : Protection code
NP : Number of pages
AB : Addresses of the first block
E : Extra

Fig. 3.5 The File Directory

CHAPTER 4

THE SYSTEM CONTROLLER

As discussed in Chapter 1, the laboratory is centred around the system controller. It is expected to execute control over different work-stations in the system, as well as over the secondary storage. For performing each of these functions different task controllers are needed such as magnetic bubble memory controller and the interface controller handling the transfer of data/command between any of the work-stations and the system controller.

In the first two sections, the requirements of the system controller and its hardware are discussed. Section 4.3 gives the command execution procedure and flow charts for the execution of different commands.

4.1 SYSTEM REQUIREMENTS

In the multiuser multiterminal environment discussed in Chapter 1, the system controller should provide the facilities to users to create a new file in his area, to read or copy a file from his or any other user's area or to protect a file from being used by any other user. Also it should be possible to find out the files in different users area and a provision to send message to the other user working on any other work station.

Moreover, it is necessary to have flexibility in allocating the secondary memory space to the different users, and to keep track of the total memory space used by each user. In addition, in multiuser environment the system controller should prevent any unauthorized user from using the facilities as well as from misusing the facilities provided to any other user.

This can be achieved by maintaining a table for all relevant information concerning the authorized users.

4.1.1 The User's Table

The different entries that are necessary in the user's table are described below :

i) The name of the user : This is an identifier to the user with which the system recognises the user. In general, a name size of 12 characters is enough for identification.

ii) The password : This field contains the user's codeword so that he can prevent the facilities offered to him from being used by any other user. A word size of six characters is provided for the password.

iii) The status : Entry in this field shows that the user has logged-in or not. A single bit would have been enough, but because of the byte structure a byte has been provided for this information also.

iv) Total number of pages : Maximum amount of memory space that can be allocated to a user is 64 Kbytes i.e. 128 pages. So a single byte in this field will be enough.

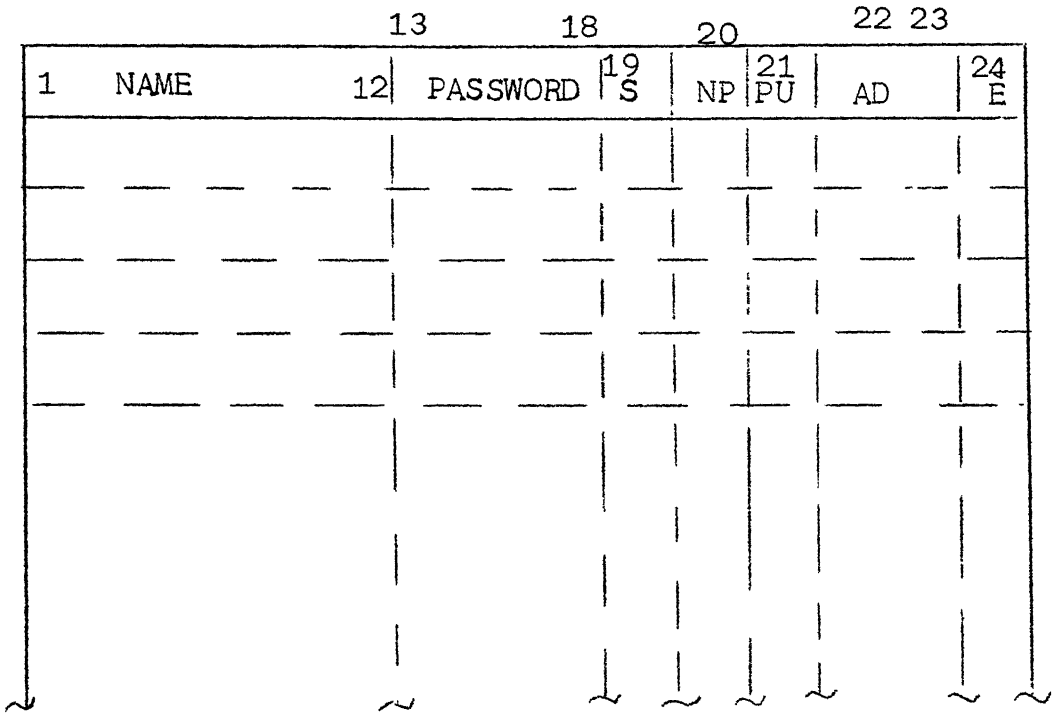
v) Number of pages used : This again requires a single byte, indicating the total number of pages user has already utilized. The number here can be atmost equal to the earlier field desired.

vi) The address of the directory : This information is required to know the location of the directory of the user. It is the address of the page in the actual memory. 2 bytes of storage is required.

From the foregoing discussion we see that 23 bytes of information have to be stored for each user. Additional one byte is left unfilled for any further modification, requiring 24 bytes per user entry in the user's table. As pointed out earlier, the system has been designed for 64 users so (64x24=) 1536 bytes of storage is required in the secondary memory. The structure of the user's table is shown in Fig. 4.1.

4.2 HARDWARE REQUIREMENTS

As seen in earlier sections, the size of the user's table, is 1536 bytes. It has to be stored in the main memory, as soon as system is switched on. Same space is also required



S : Status of the user
NP : Number of pages allocated
PU : Number of pages utilized
AD : Address of Directory
E : Extra

Fig. 4.1 The User's Table

for storing the file directory (16x64= 1024 bytes) and the block table (4x128 = 512)(refer Sec. 3.2.2) in the main memory during operation. Also during transfer of data, a data buffer of 1 Kbyte size is required. Taking all this into account and providing some additional memory for house-keeping operations, 6 Kbytes of RAM is required on the board. The other requirements are the serial interface, so that the console terminal can be connected on this link, which will always be in logged in position, making the communication between the controller and the consol always possible. In addition, the magnetic bubble memory controller is required to handle magnetic bubble memory modules.

4.2.1 System Controller Configuration

Fig. 4.2 shows the block diagram of the system controller. The controller is based on Intel's 8085 microprocessor. Ten Kbytes of RAM is provided using ten pieces of 1Kbyte RAM (8185). The EPROMs, used to provide the resident system program are 2716's. Three such EPROMs are needed for the permanent storage of the system software. The decoder logic consists of address decoders 3205s, NAND gates 7400s, and OR gates 7432s. The serial link is provided using Intel's 8251 USART. The additional circuitry for baud rate generation and the 20 mA current loop for the TTY interface are also provided. To handle the bubble memory modules, there is a bubble memory controller chip 7220, housed on the same board.

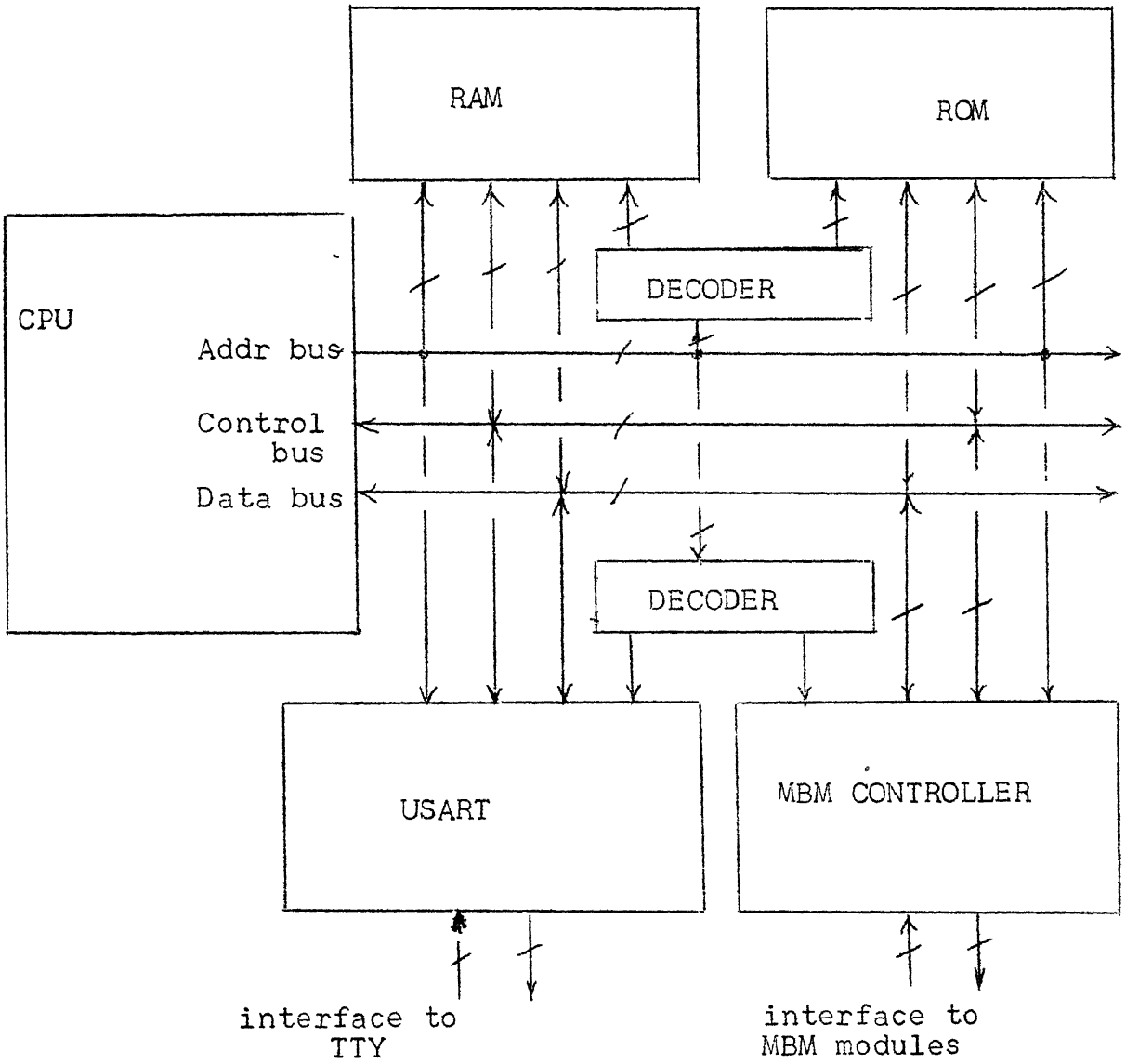


Fig. 4.2 Block diagram of the system controller

The bubble memory controller and the USART are configured as direct I/O. Additional six decoded lines are provided for expansion of the system. The signal lines from the 7220 BMC are brought on two connectors which provide interface to memory modules. The data and address bus are brought out for any further expansion. The layout for the board is given in figure 4.3.

4.2.2 Magnetic Bubble Memory Organization Scheme

The required storage capacity of 1Mbyte is obtained from 8 128Kbyte MB modules (7110) in a byte-organized configuration as shown schematically in Fig. 4.4. The bits from different magnetic bubble modules are grouped together by employing time-division multiplexing for full exploitation of the access speed. Subject to the maximum bit rate of 100Kbits/sec. for each module, this scheme permits a data transfer rate of 100K bytes/sec.

4.3 COMMAND EXECUTION PROCEDURE

The first step in the execution of any command is to decode it. After receiving the command the controller executes the command-decoding program. The commands have to be in exact formats as described in Table 4.1 and only the first three letters are considered. If the given command is not from the set of commands, the system sends a message to the user and returns back to the monitor. Once the command is decoded, the control of the program is passed to

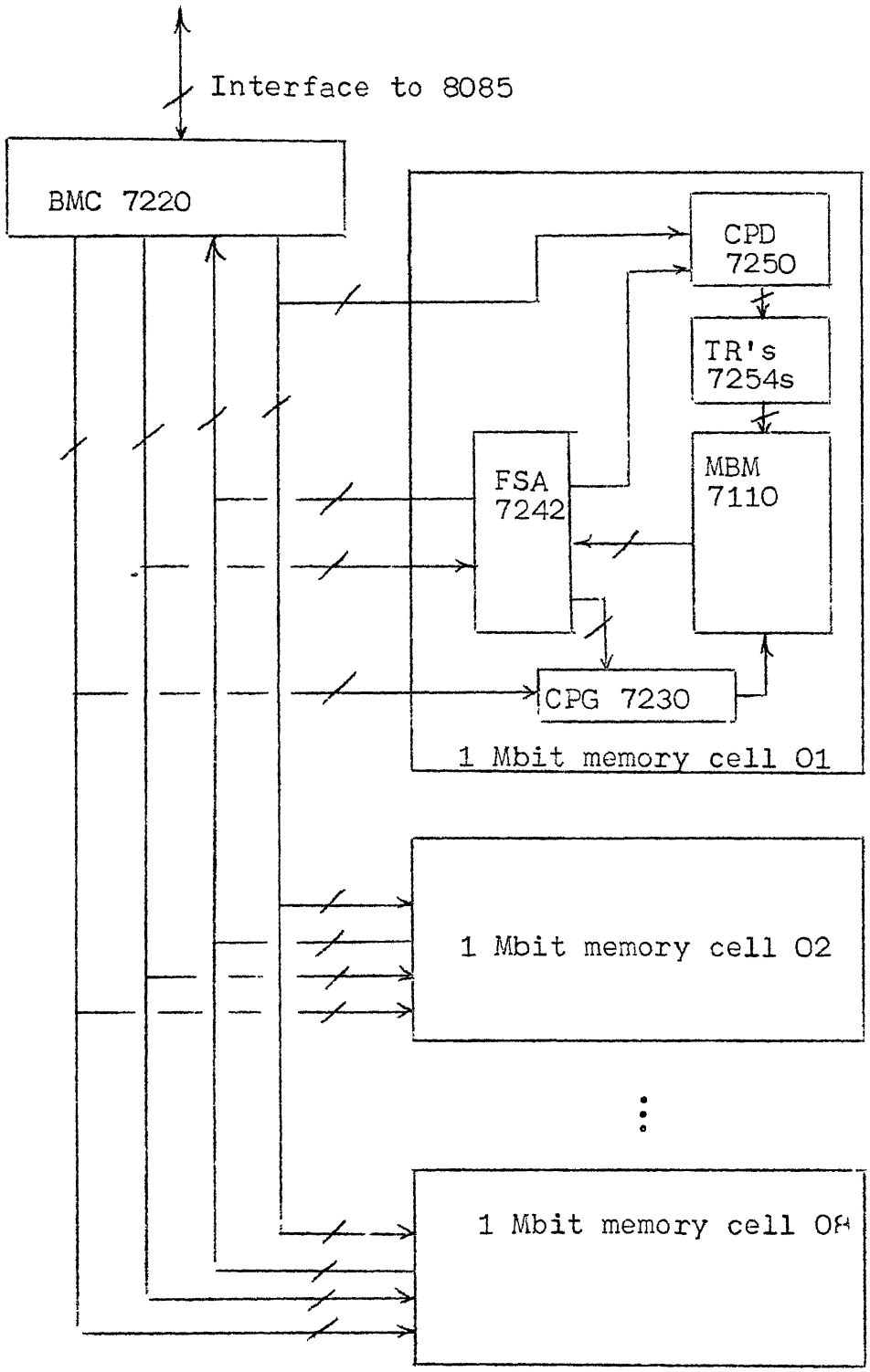


Fig. 4.4 1 Mbyte memory configuration

Table 4.1

The Command Table

	Command Format	Function
1.	LOG/name	Logs in the valid user
2.	IDE	Identifies the user on the terminal
3.	a) DIR	Types one's own directory
	b) DIR/UN	Types directory of the specified user
4.	a) REA:FN	Reads the specified file from one's own area
	b) REA UN:FN	Reads the specified (unprotected) file from the specified users area
5.	WRI FN	Writes file in one's own area
6.	REN/SFN=DFN	Changes the source file name to the destination file name
7.	a) COP:SFN=DFN	Copies the specified file from one's own area
	b) COP UN:SFN=DFN	Copies the specified (unprotected) file from the specified user's area
8.	DEL FN	Deletes the specified file from one's own area
9.	PRO FN/PC	Changes protection code of the specified file
10.	MES TN: message	Sends the given message to the specified terminal
11.	KIL	Logs off the user from the system
12.	SYS	Displays the current status of all the terminals in the system
13.	USE	Displays all the valid user's name in the system
14.	HLP	Displays this text

UN : User's name ; FN : File name ; SFN : Source file name
DFN: Destination file name ; PC : Protection code
TN : Terminal number

the respective program module which handles the command. The flow charts for the different command execution-programs are given in Fig. 4.5 to Fig. 4.15 in the following order.

i)	LOGIN	Fig. 4.5
ii)	IDENTITY	Fig. 4.6
iii)	DIRECTORY	Fig. 4.7
iv)	READ	Fig. 4.8
v)	WRITE	Fig. 4.9
vi)	RENAME	Fig. 4.10
vii)	COPY	Fig. 4.11
viii)	DELETE	Fig. 4.12
ix)	PROTECTION	Fig. 4.13
x)	MESSAGE	Fig. 4.14
xi)	KILL	Fig. 4.15

COMMAND : LOGIN

FORMAT : LOG/name

FUNCTION : Loggs in the valid user

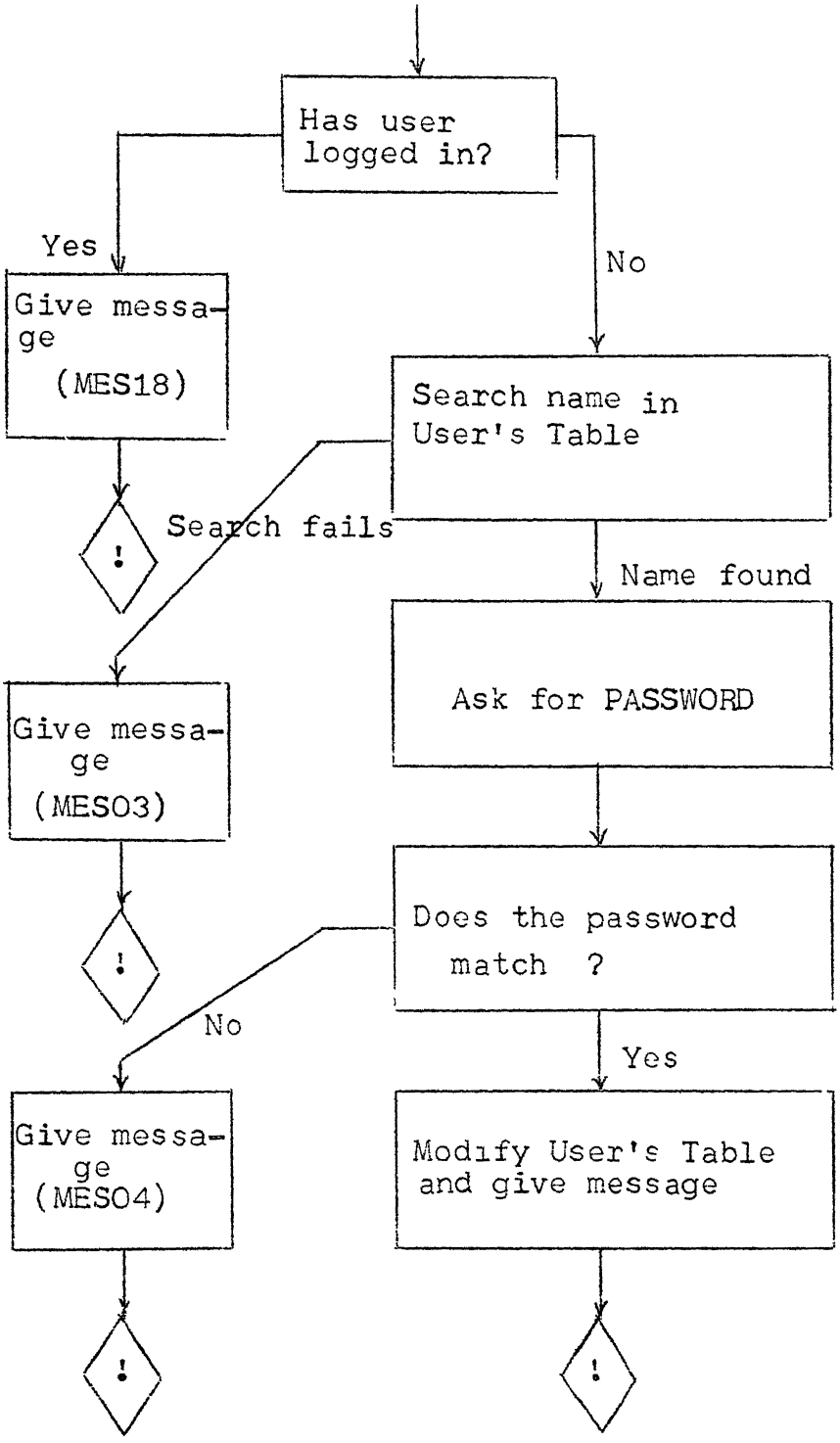


Fig. 4.5 Flow chart of command LOGIN

COMMAND : IDENTITY
FORMAT : IDE
FUNCTION : Identifies the User on the terminal

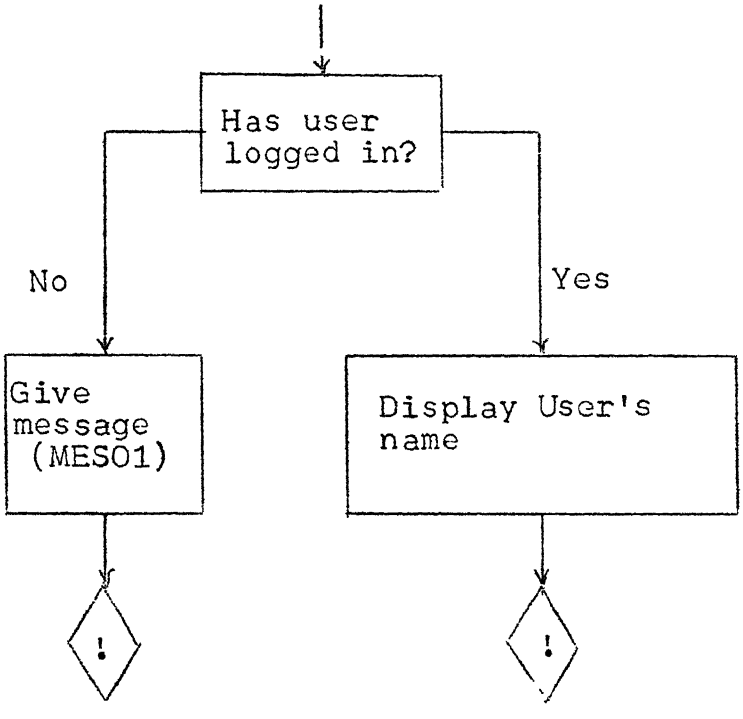


Fig. 4.6 Flow chart of command IDENTITY

COMMAND : DIRECTORY (two options)

FORMAT : (a) DIR
(b) DIR/User's name

FUNCTION : (a) TYPES one's own directory
(b) Types the directory of the specified user

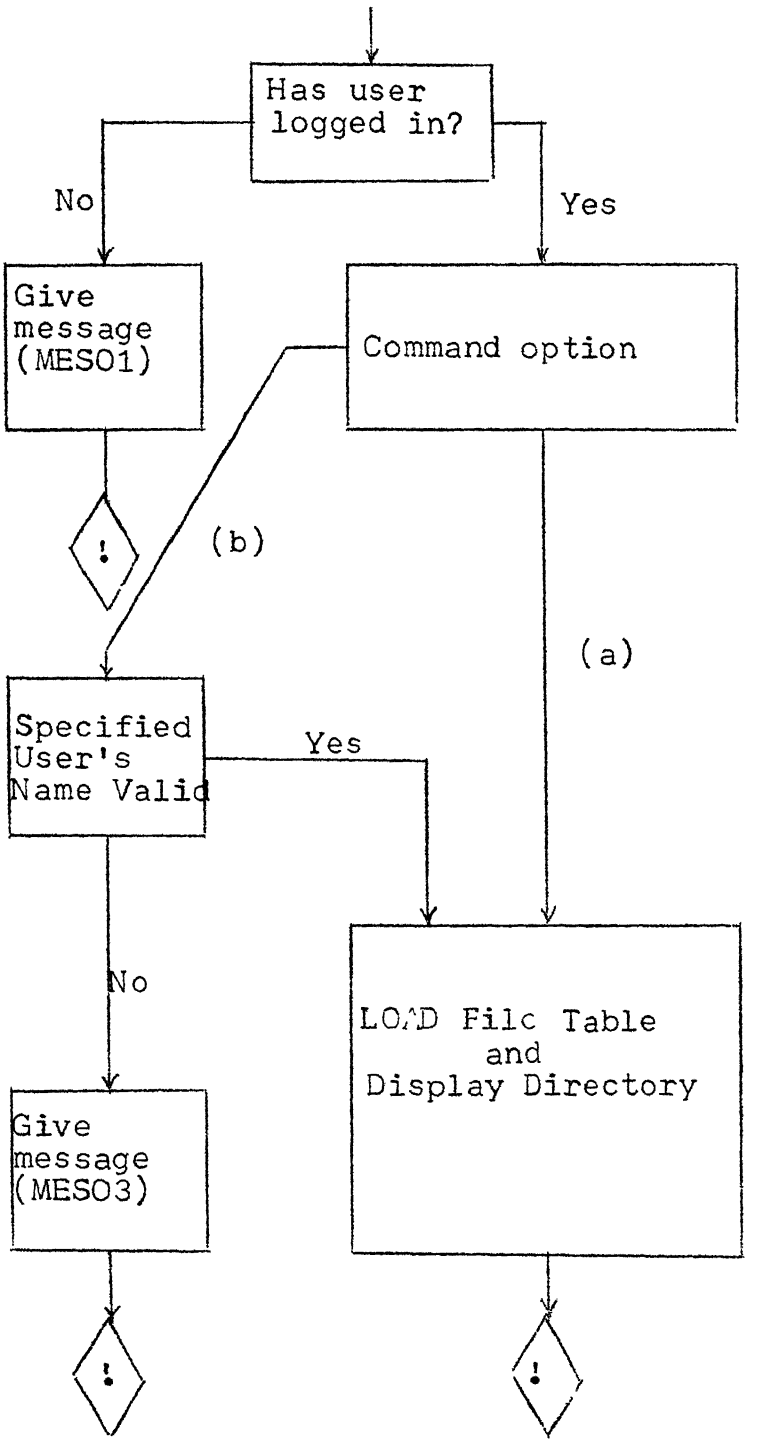
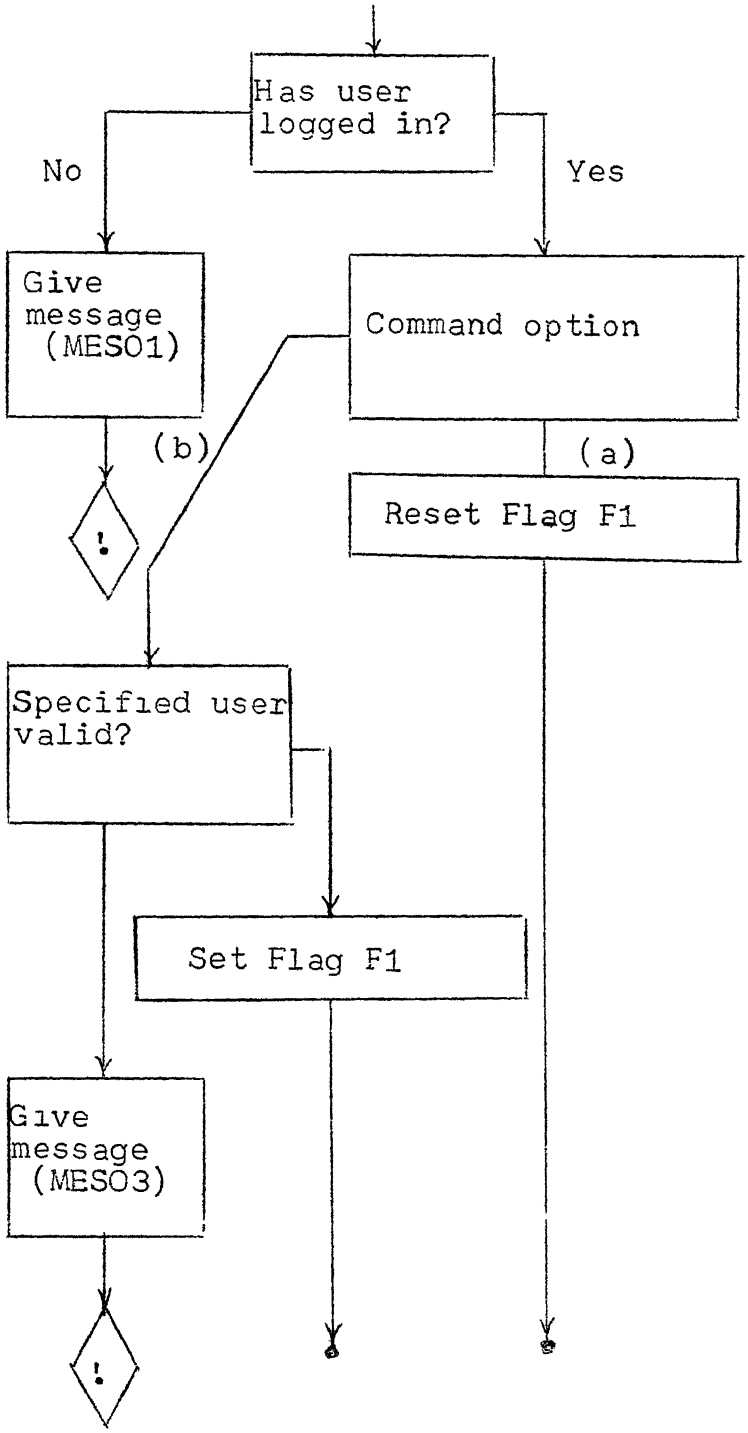


Fig. 4.7 Flow chart of command DIRECTORY

COMMAND : READ (two options)

FORMAT : (a) REA: file name
(b) REA User's name : file name

FUNCTION : (a) Reads the specified file from one's own area
(b) Reads the specified (unprotected) file from the specified user's area



contd...

contd...(P/57)

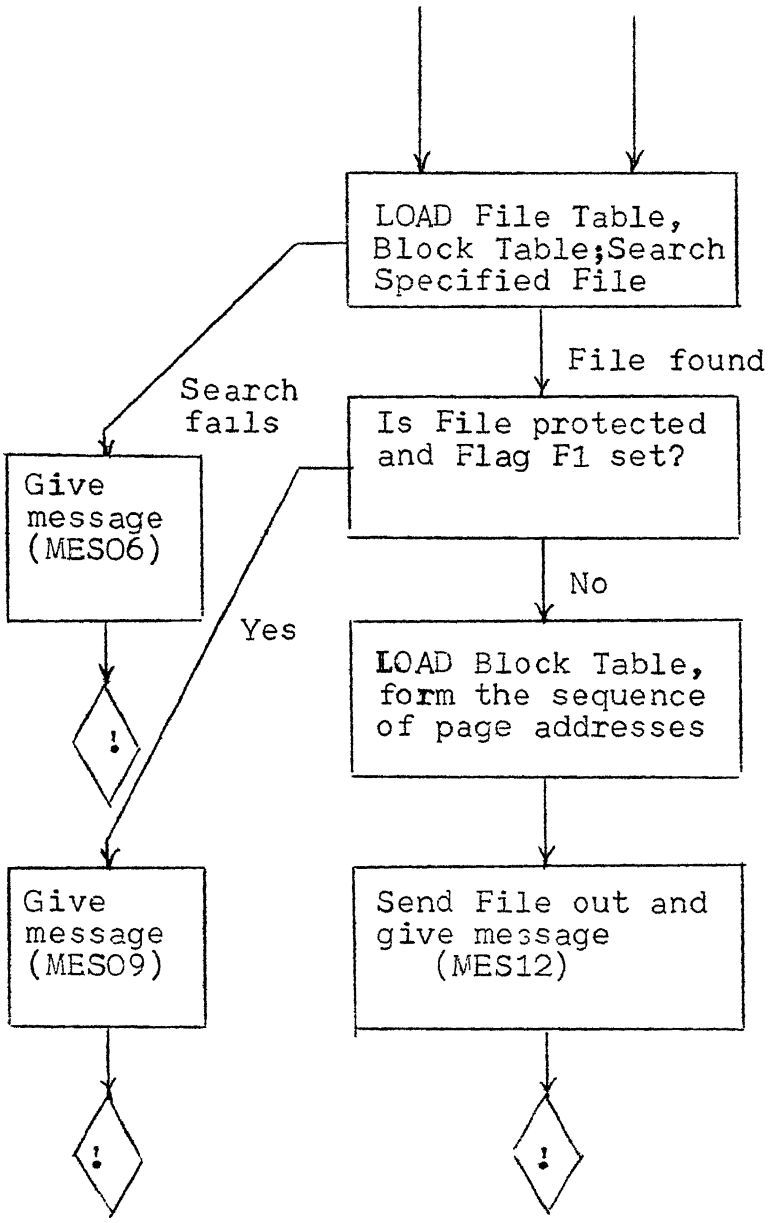


Fig. 4.8 Flow chart of command READ

COMMAND : WRITE

FORMAT : WRI file name : pages

FUNCTION : Writes the file in one's own area

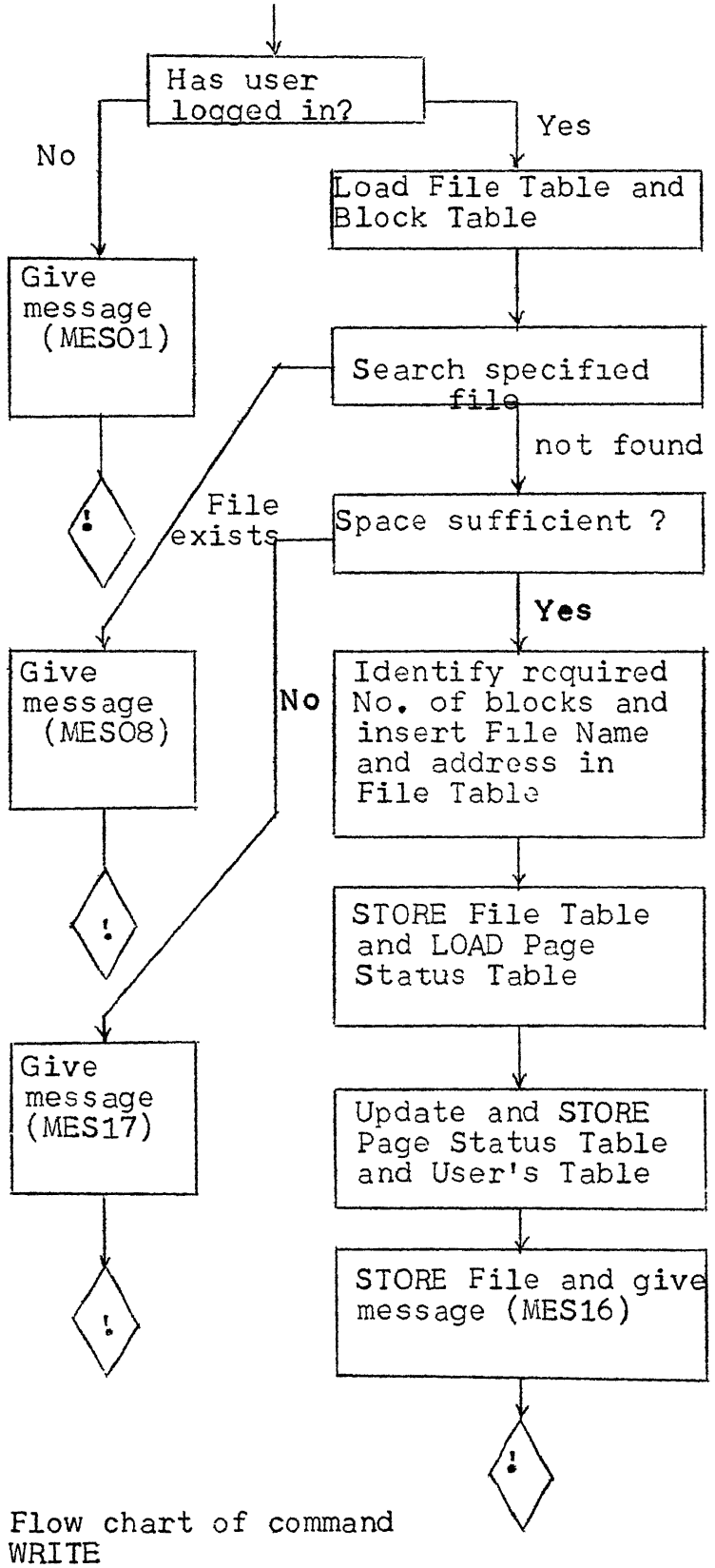


Fig. 4.9 Flow chart of command WRITE

COMMAND : RENAME

FORMAT : REN/source file name = destination file name

FUNCTION : Changes the source file name to destination file name

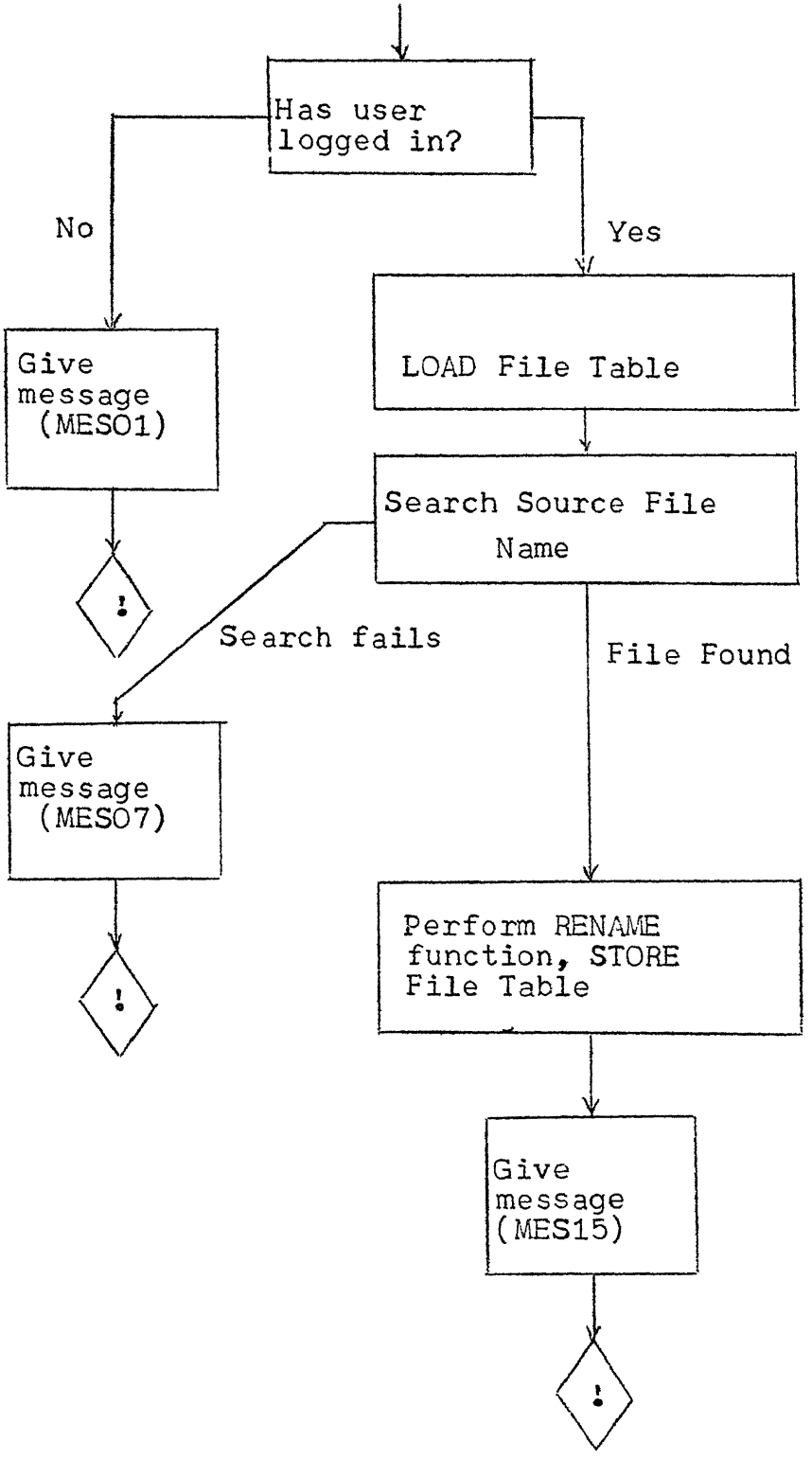
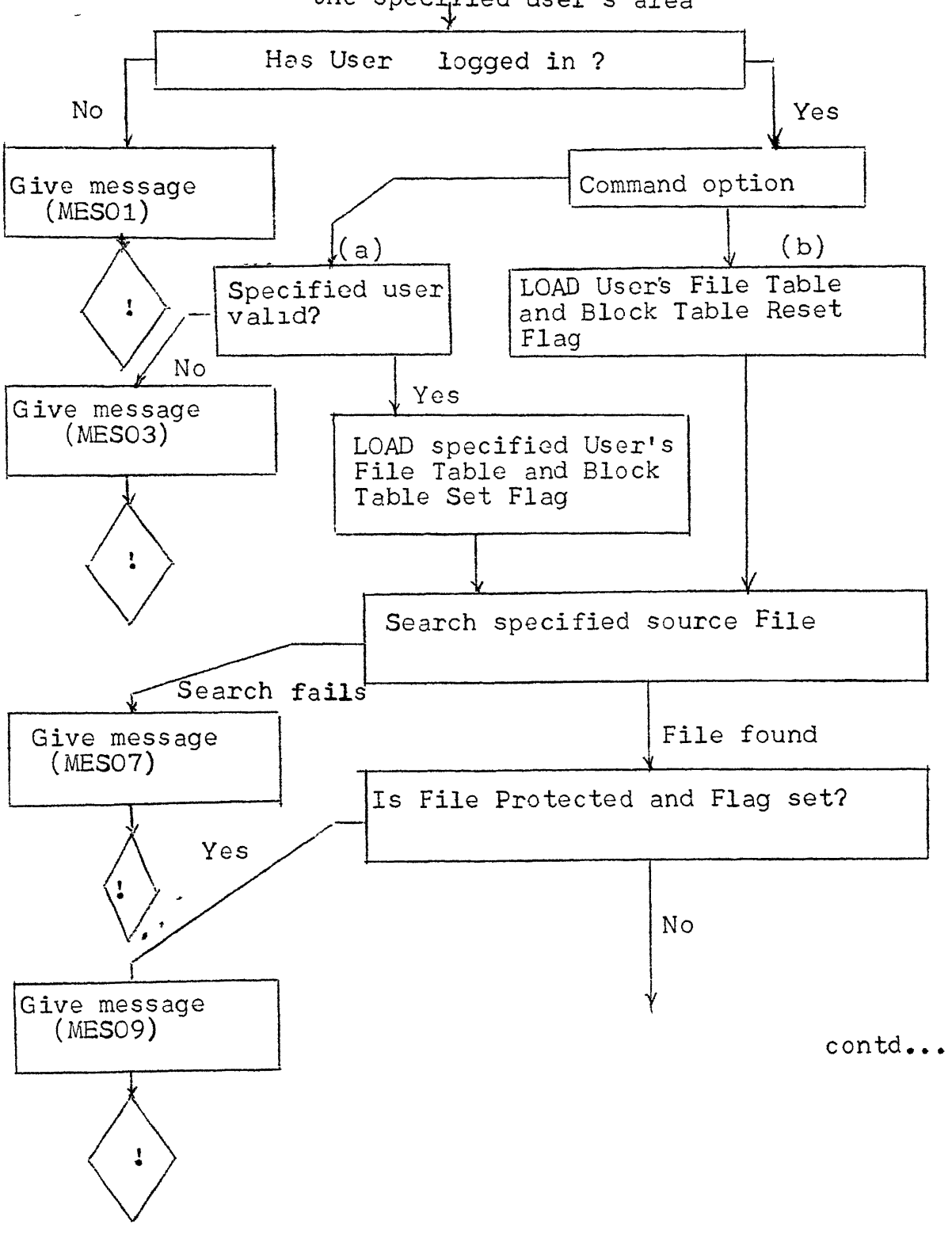


Fig. 4.10 Flow chart of command RENAME

COMMAND : Copy (two options)

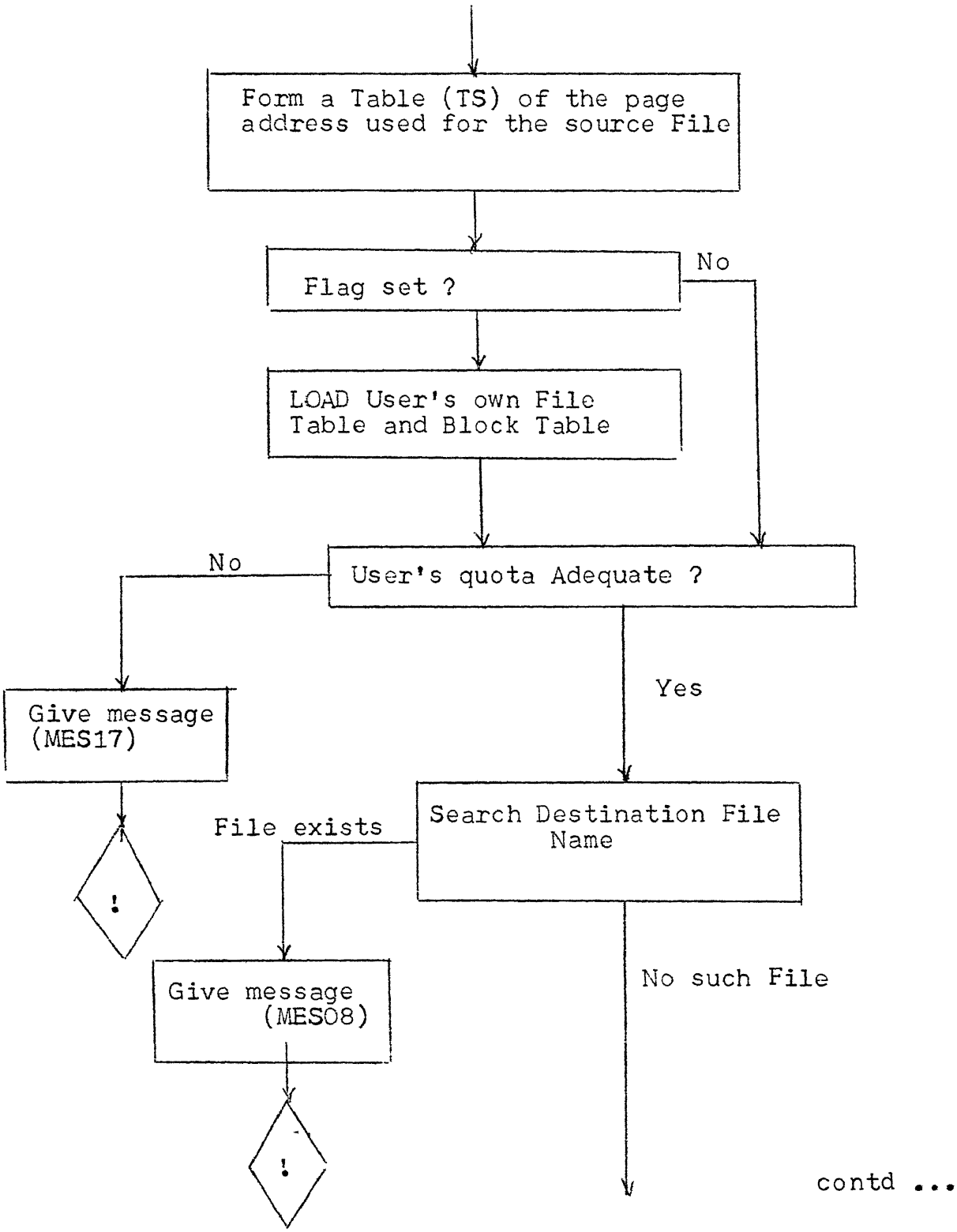
FORMAT : (a) COP : source file name = destination file name
(b) COP User's name : Source file name = destination file name

FUNCTION : (a) Copies the specified file from one's own area
(b) Copies the specified (unprotected) file from the specified user's area



contd...

contd ...(P/61)



contd ...

contd...(P/62)

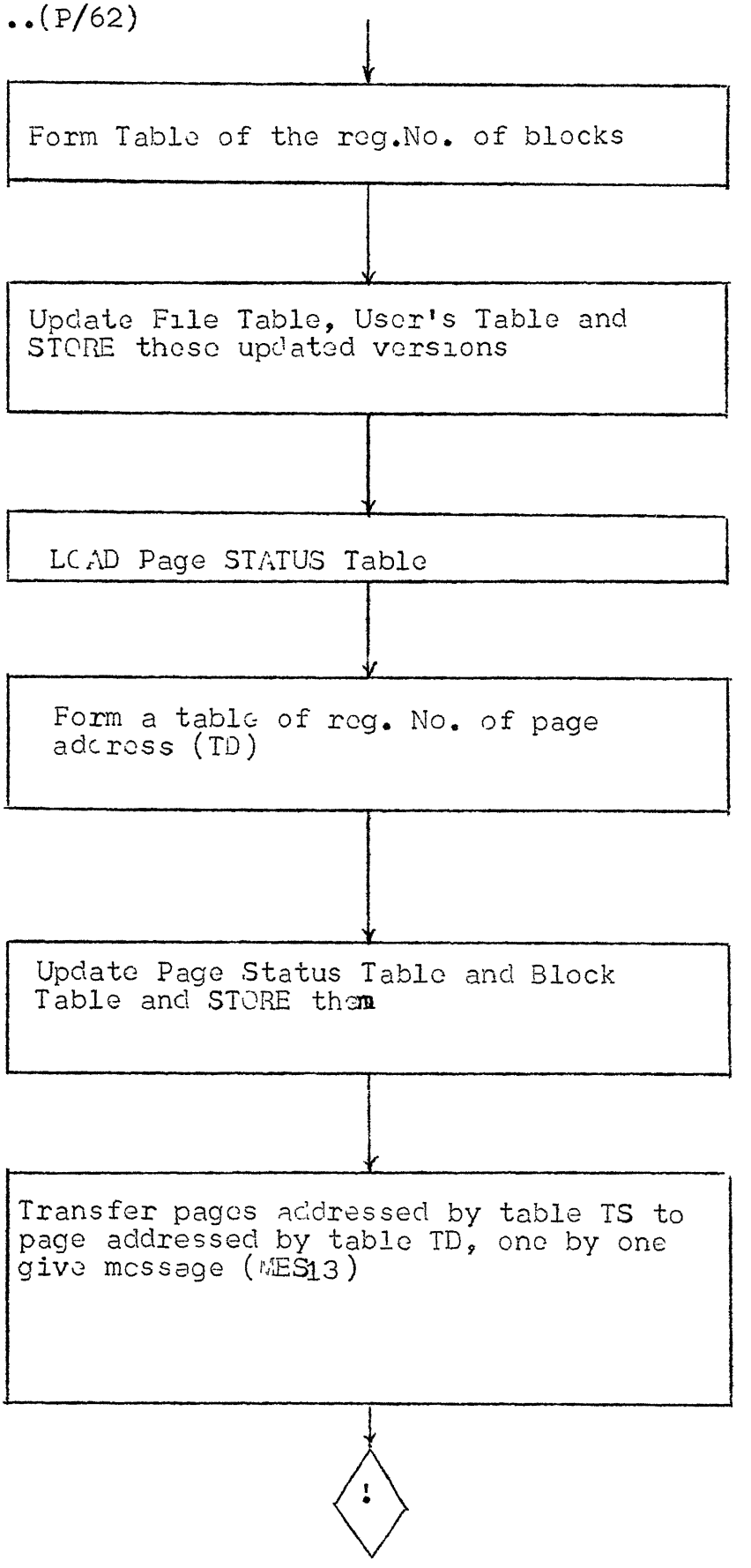


Fig. 4.11 Flow chart of command COPY

COMMAND : DELETE
FORMAT : DEL file name
FUNCTION : Deletes the specified file from one's own area

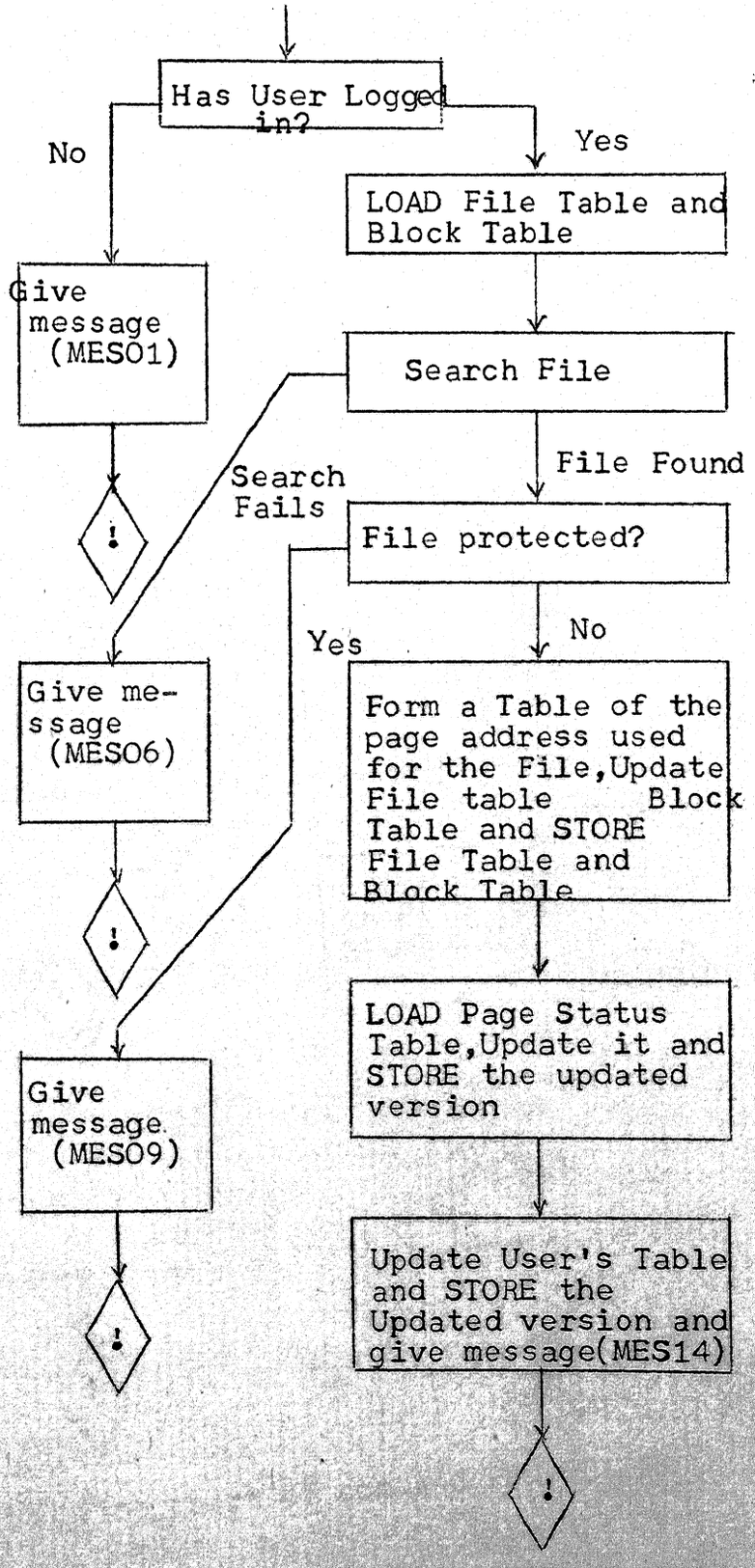


Fig. 4.12 Flow chart of command DELETE

COMMAND : PROTECTION

FORMAT : PRO file name/protection code

FUNCTION : Changes protection code of the specified file

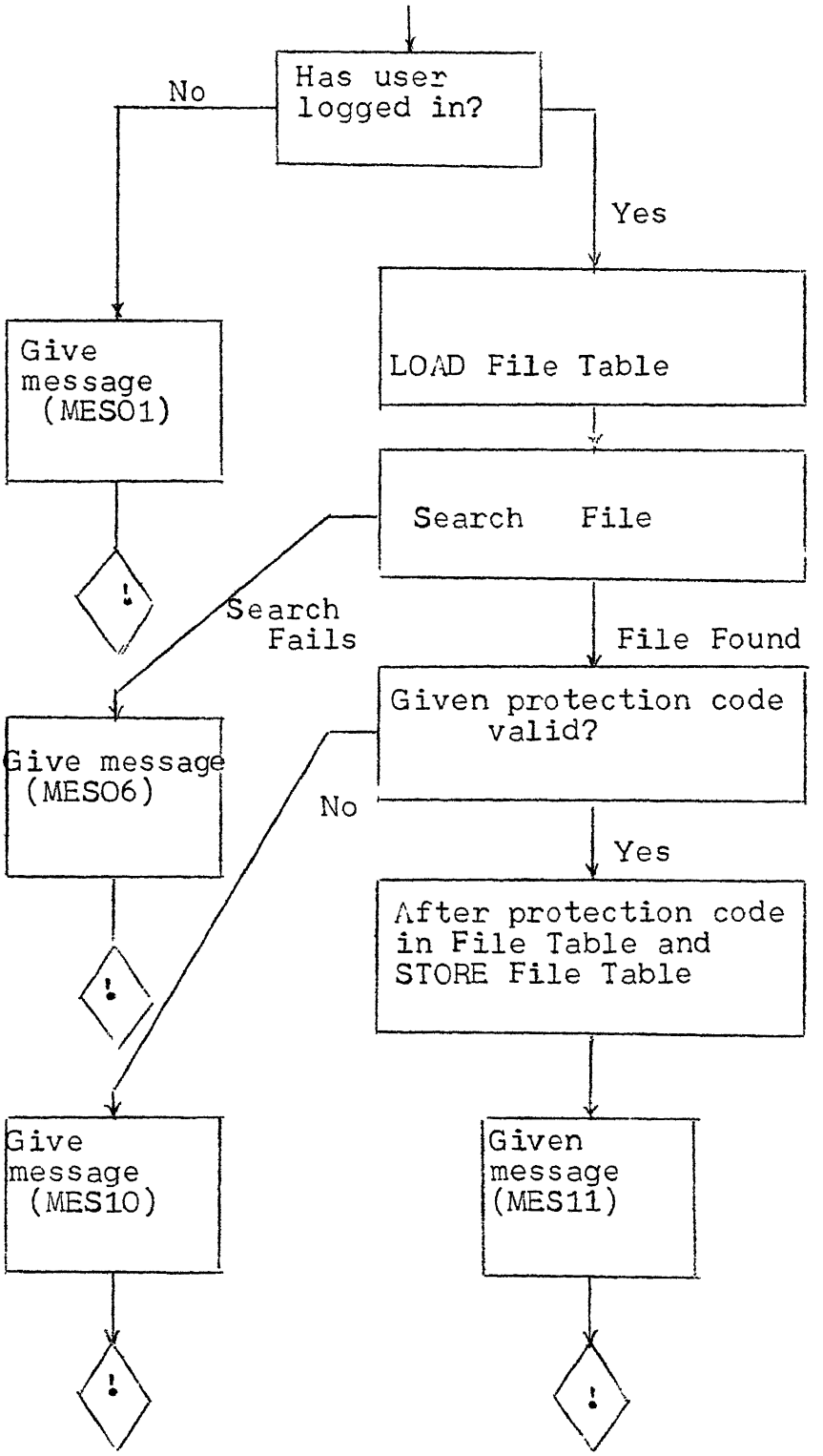


Fig. 4.13 Flow chart of command PROTECTION

COMMAND : MESSAGE

FORMAT : MES TN : message

FUNCTION : Sends the given message to th
to the specified terminal

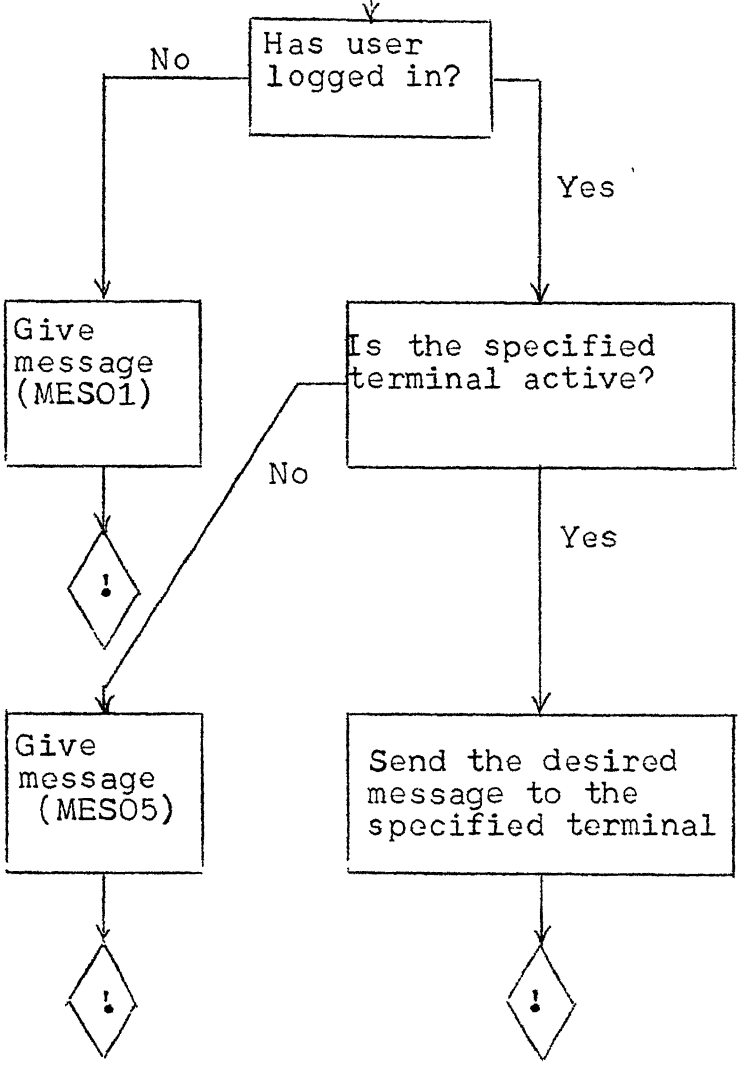


Fig. 4.14 Flow chart of command MESSAGE

COMMAND : KILL

FORMAT : KIL

FUNCTION : Logs off the User form the system

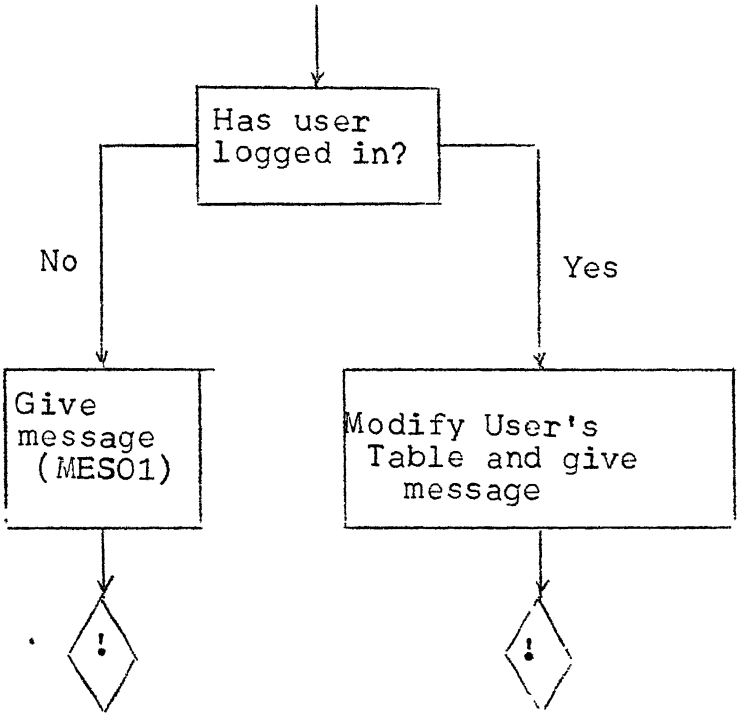


Fig. 4.15 Flow chart of command KILL

CHAPTER 5

THE CODING SCHEME

A multiterminal environment like the one envisaged in this project calls for communication links between the terminals and the secondary storage through controller. As the laboratory organization is visualized to be quite compact (16 terminals) any one of the various available techniques for base-band data transmission can be employed. The dominant mode of error in all such transmission links is the 'burst error', which involves errors of large number of consecutive bits . often due to electrical disturbances. One way to take care of this type of errors is to use a suitable 'coding' scheme together with some arrangement to distribute this cluster of errors. The purpose of coding is to introduce redundant bits into the message sequence so that the original message sequence can be recovered from the received erroneous sequence. The important factorsin deciding coding scheme are the size of the burst and the repetition rate of the burst.

5.1 PRINCIPLE OF CODING

In any practical coding scheme, the redundant 'parity' bits are introduced in an efficient way so that each possible combination of errors in the code-word results in a distinct indicator (called syndrome) at the receiver end, thereby making it possible to correct errors in any one or more bits.

The function of the encoder is to accept a k-bit message and convert it into an n-bit codeword based on a known transformation thus introducing r(=n-k) parity bits. The distance between two codewords is defined as the number of locations at which the corresponding bits differ. The efficiency of a coding scheme is given by the minimum value of the distance between all possible pairs of codewords. In order to correct t errors, this minimal distance must be atleast (2t+1). The so called 'maximal minimal distance codes' employ the maximum value of this minimal distance, and as such are the most efficient codes. For any given (say t bits) error correcting capability, the required number r of parity bits is also minimum for maximal-minimal distance codes. For these codes r and t are related by expression

$$r = 2t$$

Thus to correct single (1-bit) errors, the maximal minimal distance codes require 2 parity bits. There are no non-trivial maximal-minimal-distance binary block codes [15]. So, any nontrivial binary block code for single error correcting capability must have atleast 3 parity bits.

5.1.1 Basis of Coding

Let a message vector M and a code vector C be defined as follows :

$$M = (m_1, m_2, \ldots, m_k)$$

and

$$C = (c_1, c_2, \ldots, c_n)$$

where $m_1, m_2, \ldots, m_k$ are the k-message bits and $c_1, c_2, \ldots, c_n$ are the n bits of the codeword. The coding process can be looked upon as a matrix multiplication

$$C = M.G \qquad (5.1)$$

where G is a generating matrix of size kxn. The rows of G being given by linear combinations of a generating polynomial, g(x), which is the product of irreducible polynomials in general. For maximal-minimal-distance codes, g(x) is an irreducible polynomial itself . For the (7,4) Hamming code, two possible generating polynomials are (x^3+x+1) and (x^3+x^2+1).

5.1.2 Basis of Decoding

Let R be the received codeword, which has been corrupted during transmission :

$$R = C \oplus E \qquad (5.2)$$

where $\oplus$ indicates bit-wise modulo-2 addition and $E = (e_1, \ldots, e_n)$ is the error vector, i.e. $e_i = o$ if there is no error in the ith bit and $e_i = 1$ if the ith bit is erroneous.

The decoding scheme starts with finding the syndrome vector S which is obtained by the matrix multiplication of a parity

check matrix H with received word R

$$\text{i.e. } S = RH^T \tag{5.3}$$

If all possible errors are to be corrected, then the matrix H has to be so chosen that there is a one to one correspondence between the distinct values of the error vector E and those of the syndrome vectors S.

From eqn. (5.2) and eqn. (5.3), it is easy to see that

$$S = CH^T + EH^T = MGH^T + EH^T \tag{5.4}$$

As S should only depend on E and not on M, it follows that

$$G\,H^T = 0 \tag{5.5}$$

and hence

$$S = EH^T \tag{5.6}$$

Thus the received word is the correct codeword, if S = 0. As in the case of the generating matrix G, the rows of the parity-check matrix H are obtained by the linear combinations of the parity-check polynomial h(x), of degree k, which is the conjugate polynomial of g(x). This shows that the order of the parity-check matrix H is (n-k) xn), i.e. (rxn). This gives the size of the syndrome vector to be (1xr).

Equation (5.6) shows that if there is a single-bit error in the ith position of the received word, the syndrome is simply the ith row of the parity-check matrix. Also if

there are two errors in the ith and jth positions, the syndrome vector is the modulo-2 sum of the ith and jth rows of the parity-check matrix, and so on. The decoding strategy is to calculate the syndrome and identify the nature of error(s) (single or multiple) and flip the respective bits in the received word.

If all rows of the H matrix are nonzero and distinct then the zero-error sequence and all single-error sequences will have different syndromes and thus decoder will be capable of correcting all single errors. For an (n,k) code, there are $2^{n-k}-1$ different nonzero sequences that can be chosen as rows of H and if $n \leqslant 2^r-1$, the rows of H can be chosen to be nonzero and distinct.

5.2 HAMMING CODES

Hamming codes are binary block codes for which the rows of H are distinct and includes all nonzero sequences of length (n-k). For such codes n and k are related as

$$\begin{aligned} n &= 2^{n-k}-1 \\ &= 2^r-1 \end{aligned} \tag{5.7}$$

Some of the combinations of n,k and r are given in Table 5.1.

Hamming codes are the only binary perfect codes with single-error correcting capability [15].

Table 5.1

r	n	k
2	3	1
3	7	4
4	15	11
5	31	26

The (7,4) Hamming code has 3 parity-check bits, which is also the minimum number of parity-check bits for any single error-correcting nontrival binary block code. In this sense, the (7,4) Hamming code is the nearest to the single-error correcting maximal-minimal-distance codes.

5.2.1 (8,4) Modified Hamming Code

Since most microprocessors are byte-oriented, i.e., the data is in 8 bits, the code parameters n and k must be preferably in powers of 2 for ease in processing. Also the coding scheme must be simple enough as to make decoding realizable. The nearest Hamming code satisfying all these requirements is the (7,4) one. This can be modified into an (8,4) block code, to make it compatible with the byte structure, by incorporating an additional check bit obtained by a straight-forward parity check on all seven bits of the

original (7,4) codeword

$$\text{i.e.,} \quad c_7 = \sum_{i=0}^{6} c_i \tag{5.8}$$

The minimum distance of the code is 4, indicating that the code can detect any double error in the codeword but cannot correct it [14]. The resulting generating matrix G for the basic (8,4) code is given by

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and the parity check matrix H is given by

$$H^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

If all possible adjacent double errors are examined in the codeword defined by the above parity-check matrix H, it is seen that identical syndromes are generated by errors in bits 1 and 2 as in 7 and 8; 2 and 3 as in 6 and 7; 3 and 4 as in 5 and 6. These ambiguities can be minimized by rearranging the message and the parity bits in the codeword so that as far as possible different syndromes are generated for all possible adjacent errors. For one such arrangement the generating matrix is

$$G' = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and corresponding parity check matrix is

$$H'^{T} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

This is an optimum arrangement found by computer search in the sense that the only ambiguity which arises between adjacent errors is between those in bits 1 and 2, and 4 and 5 [16].

This modified (8,4) Hamming code is attractive in two ways :

i) It has the byte structure and thus it is suitable from the point of view of implementation.

ii) In addition to the single-error correcting capability of the original (7,4) Hamming code it can also correct adjacent double errors. As shown in Sec. 5.3 , this is particularly useful to correct the occasional bursts of longer duration.

5.3 INTERLEAVING

When errors occur in bursts one such burst event may completely corrupt a group of successive bits while bits outside the burst remain relatively unaffected. In such situations a block code which could cope with random errors would prove to be ineffective.

However, a block code can be used in this situation if interleaving is employed. The principle of interleaving is to distribute the bits of any code-word over a period of time such that no two bits of the same code-word are separated by less than a certain time interval (The interleaving time) which is usually taken to be burst duration. This ensures a significant reduction in probability that two or more bits of the same code-word will be caught in a single burst event.

Though the interleaving time is taken to be the burst duration it is possible that the burst may extend beyond the interleaving time. After deinterleaving this results in adjacent double errors and hence the capability of the (8,4) modified Hamming code to correct adjacent double errors proves to be very effective.

5.4 THE DETAILS OF IMPLEMENTATION

The coding scheme should be capable of correcting burst errors having lengths equal to the durations of the spikes associated with electrical disturbances encountered in

the environment of the visualized microprocessor laboratory. Such disturbances (e.g. due to transients on the power line, normally have durations not exceeding 1 msec. As the memory organization scheme proposed in Sec. 4.2 is planned for a maximum possible bit rate of 800 Kbits/sec., the number of bits corrupted by a burst error may be upto 800. The coding scheme to be used must have an error-correcting capability exceeding this , i.e., interleaving has to be done to such an extent as to separate any two bits of a codeword at least by 400 bits, taking into account the adjacent double error correcting capability of the modified Hamming code.

5.4.1 Encoding

The modified (8,4) Hamming encoder takes a block of 4 message bits and converts it into an 8-bit codeword with the help of the generating matrix G (eqn. 5.11). Thus, the 16 possible message words are uniquely mapped into 16 codewords from the space containing 256 words (of 8 bit length). The codewords corresponding to all possible message blocks of length 4 are given in Table 5.2.

Let X_2X_1 be the information byte. It is transformed into two codewords, say Y_4Y_3 and Y_2Y_1, by dividing the information byte X_2X_1 into two bytes $0X_1$ and $0X_2$ and then each byte (i.e. $0X_1$ and $0X_2$) is mapped into corresponding codewords Y_4Y_3 and Y_2Y_1 respectively with the help of the

look-up table. Thus for a single byte X_2X_1, two codewords Y_2Y_1 and Y_4Y_3 are generated.

$X_1, X_2, Y_1, Y_2, Y_3, Y_4$ are hex digits.

Table 5.2

Message block in binary	Code-word in binary
0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 1	0 1 0 1 0 0 1 1
0 0 1 0	1 1 0 0 0 1 0 1
0 0 1 1	1 0 0 1 0 1 1 0
0 1 0 0	1 1 0 1 1 0 0 0
0 1 0 1	1 0 0 0 1 0 1 1
0 1 1 0	0 0 0 1 1 1 0 1
0 1 1 1	0 1 0 0 1 1 1 0
1 0 0 0	1 0 1 1 0 0 0 1
1 0 0 1	1 1 1 0 0 0 1 0
1 0 1 0	0 1 1 1 0 1 0 0
1 0 1 1	0 0 1 0 0 1 1 1
1 1 0 0	0 1 1 0 1 0 0 1
1 1 0 1	0 0 1 1 1 0 1 0
1 1 1 0	0 1 0 1 1 1 0 0
1 1 1 1	1 1 1 1 1 1 1 1

5.4.2 Interleaving

To separate any two bits of a codeword at least by a distance of 400 bits, 400 codewords have to be interleaved

For ease in implementation, it has been decide to interleave 512 codewords instead. Such a scheme can correct a 1024 bit long burst error.

The interleaving process can be visualized by referring to Figure 5.1.

A block of 512 codewords is transformed into another block of same size. The process starts with formation of a byte of all the LSB's from consecutive codewords. Once the LSB's have been exhausted the process continues for the next higher bit till all the bits in the codewords are covered.

5.4.3 Deinterleaving and Decoding

Deinterleaving is the reverse operation of interleaving in which the bits from the consecutive 64 received bits are distributed as the corresponding bits of the codewords. Repetition of this process for 8 64-bit blocks leads to the forming of a block of 512 codewords. These are then decoded one-by-one in four steps as follows :

i) The syndrome vector is calculated for each codeword.

ii) Depending upon the syndrome vector, the appropriate bit(s) in the codeword are flipped.

iii) This 8-bit codeword is mapped back into a 4-bit message (0's are placed at the four MSB locations) using the look-up table (Table 5.2).

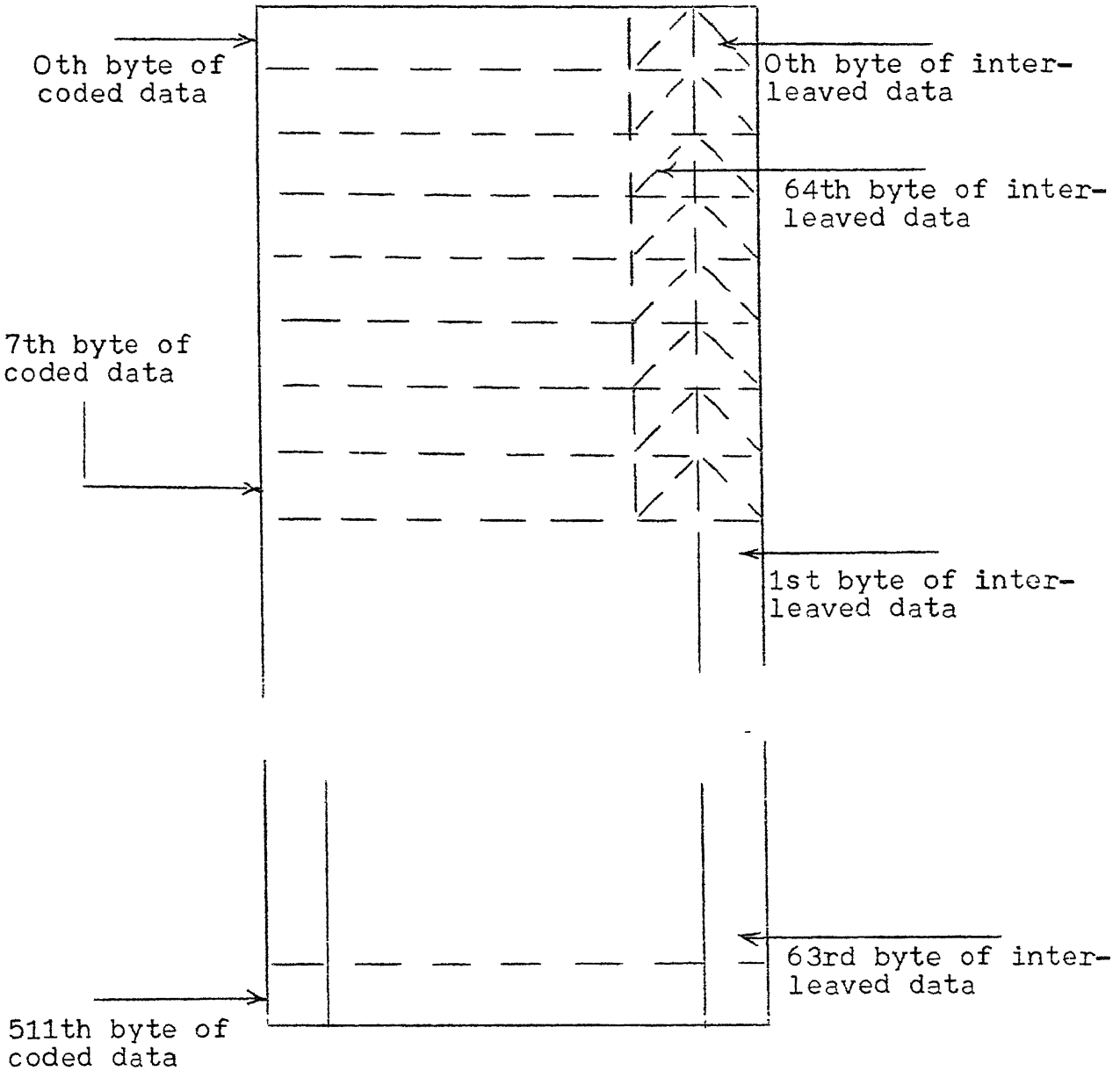


Fig. 5.1 Illustration of Interleaving Scheme

iv) If the 8-bit codeword, after flipping the bit (s), does not match with any of the valid codewords, an error flag is set; otherwise the process continues to the next codeword from the deinterleaved data until all the data is exhausted.

The decoding process ends with combining two successive decoded 4-bit messages into one byte, thus giving a block of 256 decoded information bytes.

CHAPTER 6

CONCLUSION

The system controller to handle 16 work-stations has been designed using Intel's 8085 microprocessor, as discussed in Chapters 1 and 4. The proposed scheme to manage the secondary storage has been successfully developed. The software takes care of 64 users. The limitation on the maximum number of users arises from the size of the User's table and the secondary storage requirement for storing their File and Block Tables. These limitations can be overcome, if necessary, by linking the files in the file table for a user. The software has been developed on the DEC-10 computer system and has a modular property such that any modification in the execution of a command can be carried out without affecting any other command procedure. Also the same file management software with very minor modifications can be used with any type of secondary storage like floppy disc storage etc. The software has been redesigned for 1 Mbyte of secondary storage capacity. However, with few modifications in software it can be made to handle upto 32 Mbytes of secondary storage. The total software, including the file management scheme as well as other controller commands,

requires a memory space of 6 Kbytes.

The controller board, as discussed in Chapter 4, has been designed and tested. It has provision for further expansion such as to connect bus interface controller, printer or any other peripherial device.

For 1 Mbyte to secondary storage eight magnetic bubble memory cells are required (Fig. 4.4). A printed circuit board has been designed to house two such memory cells on a single board. Four such boards are required to realize 1 Mbyte of secondary storage. The designed PCB is also modular in nature so that any number of memory cells can be attached to it.

For an initial feasibility study of magnetic Bubble memory devices as secondary storage, a 1 Mbit storage has been built using Intel's prototype kit BPK-72. BPK-72 is a single memory cell (Fig. 4.4) with one bubble memory controller (7220).

A console terminal has been provided which can execute restricted LOCAL commands like Move, Display, Substitute, Execute etc., in addition to the commands provided to the user. Console terminal can directly read from or write into any memory location. Utilizing this facility, a multiterminal environment has been simulated using one TTY, by changing its identification from time to time.

A coding scheme has been implemented in software incorporating interleaving and utilizing facilities of the console terminal, it was successfully tested for burst errors.

Also in a simulated multiterminal environment the hardware and software that has been designed and developed for the system controller and the file management was successfully tested.

From the study carried out it can be said that, it is possible to make 1 Mbyte of secondary storage using magnetic bubble memory devices with relative ease and the complexity involved in interfacing it is much less compared to any other secondary storage devices. The study also shows that the Magnetic Bubble Memory is an attractive alternative secondary storage device for an environment like a microprocessor laboratory.

REFERENCES

1. Bernard, J. Carey, A Microprocessor Laboratory for a University Environment, Computer (Jan. 1977).
2. J.E. Juliussen, D.M. Lee and G.M. Cox, Bubbles Operating First as Microprocessor Mass Storage, Electronics (Aug. 1977).
3. J.E. Juliussen, Magnetic Bubble Systems Approach - Practical Use, Computer Design, (Oct. 1978).
4. Andrew H. Bobek, Peter, I. Bonyhard and Joseph E. Geusic, Magnetic-Bubbles An Emerging New Memory Technology, Proc. of IEEE, (Aug. 1975).
5. Roman Kowalchuk et al., Magnetic-Bubble Memory, Concept, Manufacture and Testing, Circuit Manufacturing, (Sept. 1979).
6. Hsu Chang, Magnetic-Bubble Memory Technology, Marcel Dekker, Inc., New York (1978).
7. Stuart E. Madnick and John J. Donovan, Operating Systems, McGraw-Hill Kogakusha, Ltd., (1974).
8. Per Brinch Hansen, Operating System Principles, Prentice-Hall of India, New Delhi (1980).
9. Dionysios C. Tsichritzis and Philips A. Bernstein, Operating Systems, Academic Press, New York (1974).
10. Bubble Memory - Design Hand-book, Intel Magnetics, Inc. (May, 1979).

11. Magnetic Bubble Storage Data Catalog, Intel Corporation (Feb. 1981).

12. Microprocessor Interface for BPK-72, Intel's Application Note AP-119, Intel Corpor. (1981).

13. Data Catalog, Intel Corporation (1981).

14. W. Wesley Peterson and E.J. Weldon, Jr., Error-Correcting Codes, The MIT Press, (April 1978).

15. Robert G. Gallager, Information Theory and Reliable Communication, John Wiley and Sons, Inc.

16. MJD Stainforth, D.J. Watts and J.B.S. Cairns, A Forward Error Correcting Codec/Interleaver for a Digital Troposcatter System, Proc. No. 49, of IERE, London.

APPENDIX

LIST OF SYSTEM MESSAGES

MES01	Login Please
MES02	User logged off
MES03	User's name not found
MES04	Password not matching
MES05	Specified terminal not in use
MES06	File not found
MES07	Source file not found
MES08	(Destination) file already exists
MES09	File protected
MES10	Invalid protection code
MES11	Protection changed
MES12	File read
MES13	File copied
MES14	File deleted
MES15	File renamed
MES16	File written
MES17	Quota inadequate
MES18	Kill the job before logging in

EE-1982-M-DES MEG